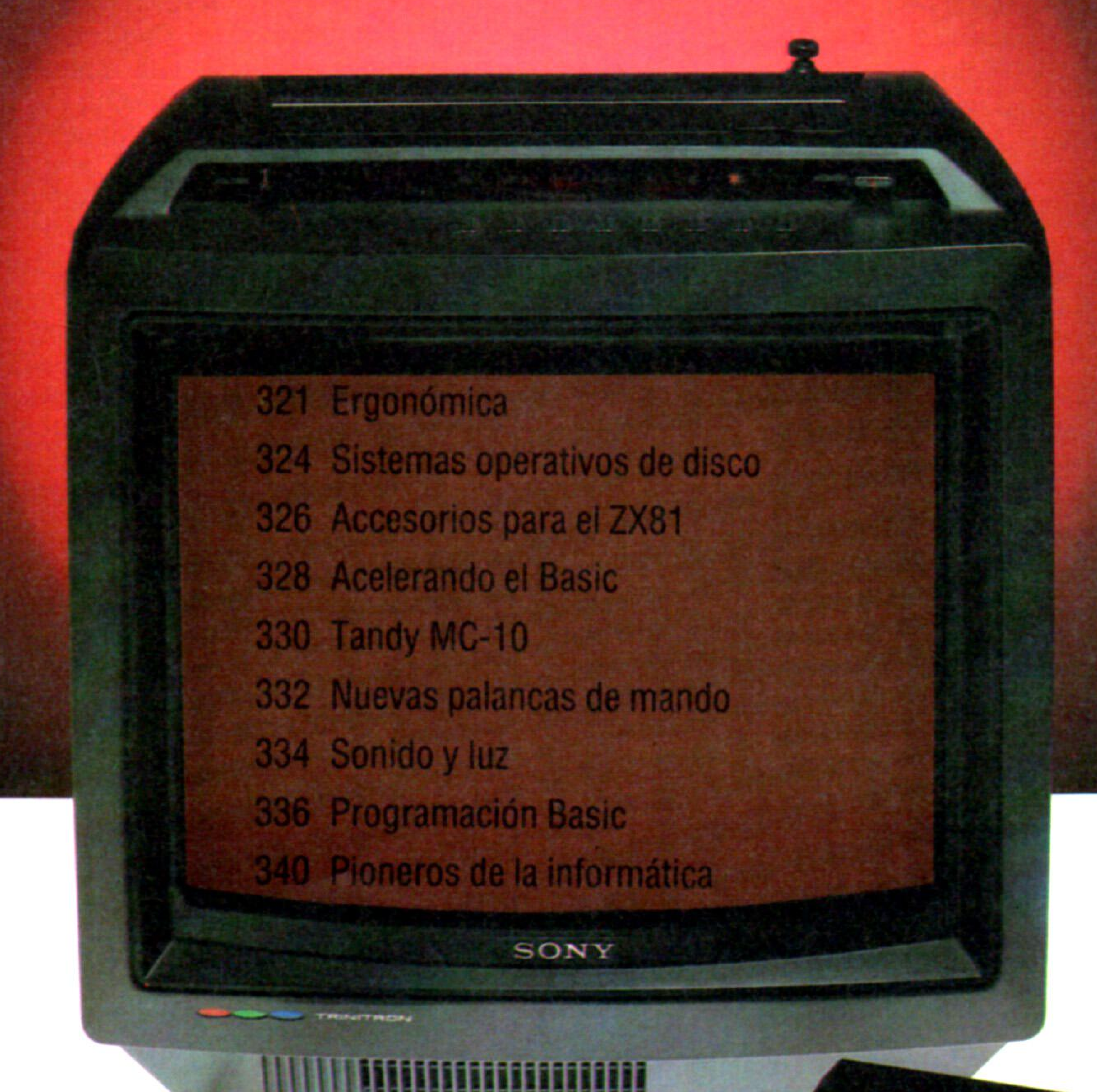
17

mi computer

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR



100017

9 788485 822836

Editorial Delta, S.A.

150 ptas.

MI COMPUTER CURSO PRACTICO

DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona, y comercializado en exclusiva por Distribuidora Olimpia, S.A., Barcelona

Volumen II - Fascículo 17

Director:	José Mas Godayol
Director editorial:	Gerardo Romero
Jefe de redacción:	Pablo Parra
Coordinación editorial:	Jaime Mardones
Asesor técnico:	Jesús Nebra

Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti, A. Cuevas
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, Barcelona-8
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador,* se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-85822-90-0 (tomo 2)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 098405
Impreso en España - Printed in Spain - Abril 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, Madrid-34.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio Blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Ferrenquín a Cruz de Candelaria, 178, Caracas, y todas sus sucursales en el interior del país.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 3371872 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Distribuidora Olimpia (Paseo de Gracia, 88, 5.°, Barcelona-8), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Distribuidora Olimpia, en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.



Manejables y atractivos

La "ergonómica" es una ciencia cuyo fin es lograr máquinas más agradables de utilizar. En el caso de los ordenadores, la investigación se ha centrado en la pantalla y en el teclado

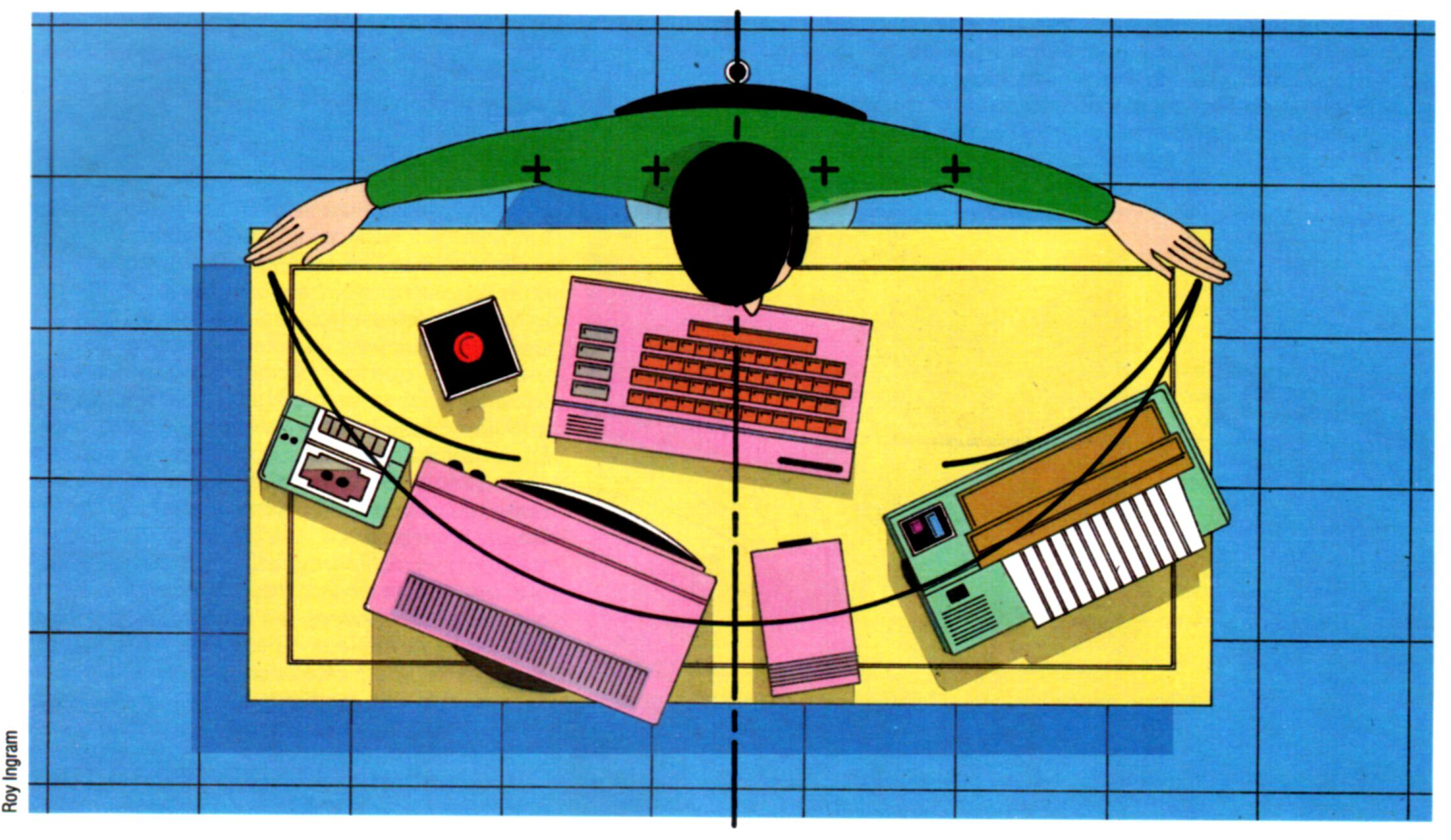
El diseño de una máquina tiene dos facetas: la estética, que se refiere a la belleza formal de su aspecto, y la ergonómica, que se ocupa de la relación existente entre el trabajador y su entorno. Independientemente del hecho de que un aparato funcione bien, no nos sentiremos a gusto usándolo si su aspecto es desagradable. Del mismo modo, el entorno en el cual estemos laborando no debe ser ni incómodo ni perturbador.

Probablemente la calidad ergonómica de una máquina, en cuanto factor a considerar en el momento de decidir qué ordenador comprar, merezca menos atención que su precio y su rendimiento. Sin embargo, vale la pena evaluar el entorno físico en el cual se va a utilizar el ordenador. En primer lugar, ¿trabaja en un sitio con el adecuado espacio libre y sobre una superficie situada a la altura correcta para usted? ¿O simplemente conecta su ordenador personal en el televisor familiar y allí se pone a trabajar, con la máquina en el regazo o, peor aún, tumbado de bruces en el suelo, frente al aparato de televisión?

La programación de ordenadores ya es de por sí lo suficientemente complicada como para que la hagamos más difícil todavía trabajando en un entorno inadecuado. Existen muchas maneras de crearse un centro de trabajo más cómodo. Comencemos por considerar lo que se puede hacer para que resulte más cómodo leer en la pantalla. Si está utilizando un televisor doméstico, entonces no podrá beneficiarse de los últimos avances tecnológicos que ayudan a reducir o que eliminan el brillo de la pantalla de los monitores. Éstos incluyen filtros para reducir al mínimo el reflejo y fósforos de pantalla coloreados especialmente. Pero usted puede mejorar la calidad de la visualización de un televisor colocando un filtro sobre la pantalla. Obtener un filtro coloreado sencillo es fácil y también se puede utilizar un filtro polarizante, que elimina los reflejos. Estos métodos ayudan a lograr un gran contraste a niveles bajos de brillo y, por consiguiente, evitan el innecesario esfuerzo de los ojos.

Los niveles de luz externa también son importantes. Cuando se trabaja de noche es mucho mejor emplear una lámpara de escritorio baja que ilumine el teclado y las notas sobre las que esté ocupado, pero que deje a la pantalla, en comparación, sumida en mayor oscuridad. La distancia desde los ojos a la pantalla también es importante; el espacio adecuado entre el cuerpo y la pantalla corresponde aproximadamente a la longitud de un brazo extendido. La visualización en sí misma debe ser inclinable, de manera que el plano de la pantalla esté a 90º respecto a la línea imaginaria que va desde su vista

El lenguaje del cuerpo
Aunque el cuerpo humano varíe en cuanto a forma y tamaño, las proporciones son siempre muy constantes, tal como comprenden enseguida los estudiantes de dibujo corporal. La ergonómica se vale de esta consecuencia lógica para definir las reglas generales del trazado de entornos de trabajo. En el caso de un ordenador personal o de unidad de representación visual, estas reglas indican que la pantalla ha de estar a una distancia equivalente a la longitud del brazo (para aminorar los cambios en la distancia focal del ojo al dirigir la vista desde la pantalla a la fuente de consulta, y viceversa). La posición del teclado obedece también a estas mismas reglas



Roy Ingram

hasta el centro de la misma. Esto se puede conseguir con facilidad colocando uno o dos libros bajo la parte delantera del aparato. No obstante, en este punto es posible que se encuentre con otro problema: su reflejo en la pantalla. Este inconveniente se puede superar de manera eficaz colocando un filtro de superficie mate.

Una vez que se ha conseguido que la pantalla resulte cómoda de utilizar, podemos pasar a considerar el trazado y las características físicas del teclado. Los factores más importantes son la altura de las teclas por encima del escritorio donde está colocado el teclado, y el ángulo de las filas de teclas entre sí. Lo ideal sería que el teclado fuera lo suficientemente bajo como para que las muñecas y los antebrazos del operador pudieran descansar sobre el escritorio, frente a él, para lo cual debería ser regulable. Lamentablemente, son muy pocos los microordenadores personales diseñados con el perfil bajo requerido. Las series ZX de Sinclair, el Oric-1 y el Jupiter Ace constituyen excepciones en este sentido, pero todos ellos plantean problemas todavía mayores con sus teclados porque, en vez de las teclas de muelle, presentan membranas de capas múltiples o láminas de plástico moldeado. Las membranas de capas múltiples no poseen sensación táctil y, en el caso del ZX80 y del 81, están espaciadas de tal modo que se hace difícil escribir al tacto con ellas. La conjugación de estas características hace que el dar entrada a programas largos sea una tarea agotadora. El Oric-1 y el Spectrum intentan subsanar este inconveniente mediante la producción de una señal audible que indica que la tecla se ha pulsado lo suficiente como para hacer contacto. Pero esta configuración dista mucho de constituir una compensación adecuada. Existen varias empresas que proporcionan teclados alternativos (a escala natural, con teclas de muelle) para los ordenadores Sinclair, pero los modelos bien diseñados son caros. También mantienen la convencional característica de poseer una entrada de tecla única —ideada por Sinclair para acelerar la operación en BASIC— bastante incómoda para el usuario.

El trazado ideal de un teclado requiere que las filas de teclas, miradas desde un costado, estén dispuestas como si formaran parte de la circunferencia de un tambor. Ello reduciría al mínimo el movimiento direccional de los dedos del mecanógrafo. Los únicos ordenadores personales que responden a esta especificación son el BBC Micro, el Commodore 64 (así como los últimos Vic-20) y el Apple II.

La máquina estenográfica
Cuando hay necesidad de grabar la voz y al taquígrafo no le es posible lograr que el orador hable con mayor lentitud, a menudo se emplea un dispositivo denominado Palantype. Las máquinas de este tipo utilizan una versión estenográfica de la grafía fonética

Martin Burke

El trazado del teclado viene siendo desde hace ya mucho tiempo la manzana de la discordia de los diseñadores. Cuando aparecieron por primera vez las máquinas de escribir, en el siglo XIX, había tantos trazados de teclado diferentes como fabricantes, pero en general las teclas de los caracteres que se utilizaban con mayor frecuencia estaban agrupadas en el centro del teclado. Cuando se introdujo la *typebasket* (máquina de escribir con las palancas de letras dispuestas en forma de cesta) en la década de 1870, los fabricantes descubrieron que hasta a los más lentos mecanógrafos se les podían trabar y enredar entre sí las palancas de letras. El problema se presentaba con mayor frecuencia cuando palabras como *ten* (formada por tres de las letras más empleadas en inglés, que se hallaban convenientemente situadas una junto a la otra en el teclado) se utilizaban en rápida sucesión. La solución adoptada consistió en desplazar aquellas letras que con más frecuencia se encontraban junto a otras en las palabras a un lugar más alejado de la *typebasket*; y así nació el teclado QWERTY, hoy estandarizado, que diseñaron Scholes y Gliden en Estados Unidos. No existe razón alguna por la cual un teclado electrónico haya de ceñirse a este trazado, como no sea para conservar una característica convencional; éste es un ejemplo de un estándar universal *de facto* que se está volviendo poco deseable y que, no obstante, es imposible modificar. Sin embargo, se han realizado algunos esfuerzos por crear teclados alternativos. En 1977, Lillian G. Malt empleó la flexibilidad inherente del hardware electrónico para producir un teclado moldeado para adaptarse a la mano, que resulta mucho más cómodo de utilizar que los de diseño estándar. Su funcionamiento también es mucho más rápido: es posible pulsar 300 palabras o más por minuto. Lamentablemente, no ha logrado imponerse al dominio completo que la disposición QWERTY posee en el trazado del teclado de los ordenadores.

Este teclado (que se llama Maltron) tiene una característica muy útil, que comparte con otros microordenadores: es desmontable. La mayoría de los ordenadores personales no poseen monitor incorporado y son lo suficientemente pequeños como para ser trasladados, pero esto no sucede con muchos microordenadores creados para ser utilizados en oficinas. Sin embargo, poco a poco se va imponiendo el diseño de teclados lo más delgados posible y que se acoplan al microordenador mediante un "cordón umbilical". El PC Junior de IBM ha dado otro paso hacia adelante: la conexión para comunicaciones entre el teclado y el microordenador es similar a los controles remotos de los televisores y videos y funciona mediante luz infrarroja.

Debido a que la ergonómica no es una ciencia totalmente objetiva (es el estudio de la forma en que los trabajadores *se relacionan* con su entorno laboral, y esa relación tiende a cambiar de cuando en cuando), no es posible dictar reglas estrictas y expeditas. Su principio fundamental es lograr un confort a largo plazo. Ello requiere que las herramientas y el equipo estén dispuestos de tal modo que uno pueda dedicar toda la energía a la tarea que tiene entre manos, sin que sea necesario cambiar de posición constantemente ni llegar a un grado excesivo de cansancio.

Existen varias posibilidades más que el usuario de un ordenador personal debe considerar y explorar con el fin de mejorar su entorno de trabajo. Cuando analizamos el Lisa de Apple (véase p. 261) observamos que al trabajar con un software activa-

Los dos teclados
Antes de que se desarrollaran los teclados electrónicos, cada tecla de la máquina de escribir se había de conectar físicamente a la diminuta pieza fundida con la forma del carácter. Esto suponía limitaciones en cuanto al trazado del teclado, ya que era esencial mantener separadas las teclas más utilizadas para que las palancas portadoras de los caracteres no chocaran entre sí. Aunque ya no es necesaria esta providencia, aún se conserva el trazado QWERTY tradicional. Los teclados como el Maltron, en el que las teclas están dispuestas de acuerdo a su frecuencia de uso, no han logrado hacerse populares

do por menú había ciertas alternativas para el teclado. Quizá le interese probar con una versión de este tipo de software, utilizando una palanca de mando o un mando de bola, y juzgar por sí mismo las ventajas. Por supuesto, necesitará escribir algunos pequeños programas con los cuales trabajar, pero empleando PEEK y POKE dentro de los límites de la memoria de pantalla no sería tarea difícil.

Por otra parte, si el ordenador que emplea permite que se vuelva a especificar el valor de una tecla determinada, podría cambiar la disposición del teclado, pegando etiquetas sobre las teclas para indicar sus nuevos valores. En este caso quizá fuera más fácil examinar (PEEK) el valor de los ocho bytes que componer el carácter en una matriz con ocho variables, cambiar los valores dentro de la matriz y después volverlos a colocar (POKE). A través de la orden POKE, podría almacenar los ocho bytes que componen el carácter directamente en el espacio destinado al carácter que desea reemplazar, pero si utiliza este procedimiento no olvide guardar el primer juego de valores en una matriz temporal y después desplazar en orden cada carácter a su nueva posición. Guarde en cassette el programa que efectúa esta operación, porque cuando apague el ordenador (o lo restaure), ¡cada carácter recuperará su valor original!

Por último, si sus intereses también se orientan hacia la carpintería, podría considerar la posibilidad de construirse un centro de trabajo diseñado a la medida con el teclado dispuesto en la parte superior y el televisor o el monitor situados en un ángulo apropiado. Las versiones comerciales de los centros de trabajo suelen proporcionar espacio adicional para el almacenamiento auxiliar (unidades de disco o cassette) en estantes situados bajo la mesa. La ergonómica se podría definir como el sentido común aplicado; si usa algunas de las sugerencias reseñadas, apreciará una significativa disminución de los dolores de cabeza y la fatiga visual.



Alternativas para el futuro
Muchos diseñadores de ordenadores prescindirían por completo de los teclados si les fuera posible. Los microordenadores más recientes, con más memoria y mayores velocidades de procesamiento, permiten utilizar, en sustitución, otros dispositivos, como palancas de mando, mandos de bola y "ratones", con el software adecuado

Kevin Jones

Sistema operativo

La función del sistema operativo de disco es no perder de vista dónde está todo cuanto se conserva en el disco. Sin él, la programación sería una tarea muy difícil

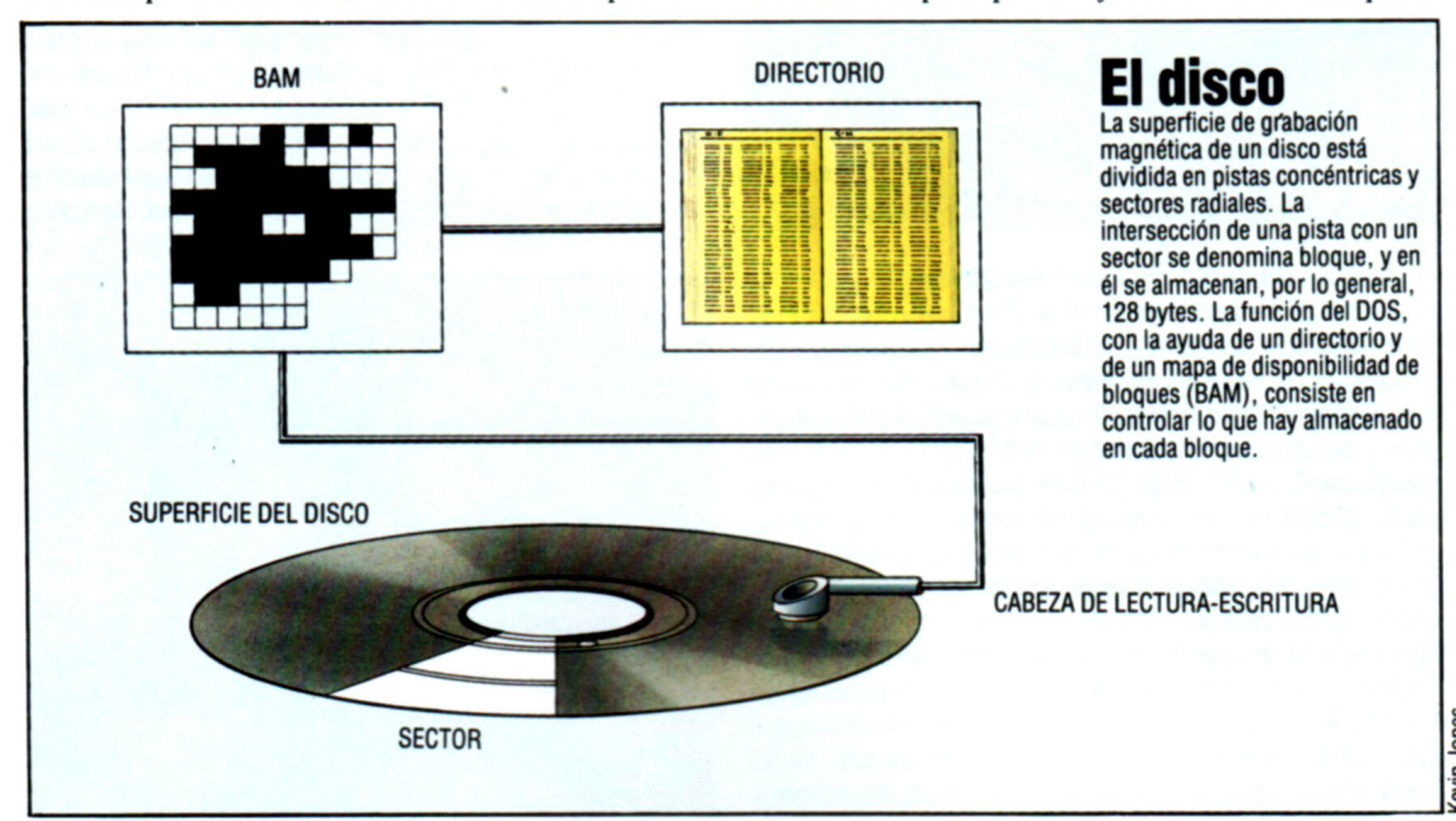
Antes de que un ordenador sea capaz de ejecutar cualquier clase de programa aplicativo, necesita su propio conjunto de programas internos con los cuales administrar los diversos componentes de su sistema e interpretar las instrucciones de que consta el programa del usuario. Este conjunto interno de programas se denomina *sistema operativo* (*Operating System*: OS), y en la mayoría de los ordenadores personales reside con carácter permanente dentro del ordenador, bajo la forma de memoria ROM.

Si su sistema incluye una unidad de disco, entonces gran parte de ese OS estará dedicada a las diversas operaciones del disco. A este conjunto de rutinas lo denominamos *sistema operativo de disco (Disk Operating System)* o DOS. Es posible que haya visto aparecer estas tres letras en los nombres de productos patentados; el sistema operativo de Microsoft, por ejemplo, se llama MSDOS. Un DOS se presenta, por lo general, en tres diferentes formas. En la primera, puede ocupar parte de la ROM interna del ordenador. Un ejemplo de ello es el Sinclair Spectrum, que lleva incorporadas las órdenes para hacer funcionar el Microdrive.

En una segunda forma, el DOS se almacena en ROM dentro de la propia unidad de disco. Esto sólo es aplicable cuando el disco es un dispositivo "inteligente" (como la unidad de disco Commodore), es decir, cuando incorpora su propio microprocesador de ROM y RAM. La fabricación de estas unidades de disco es mucho más cara, pero ofrecen considerables ventajas respecto a las unidades de disco "no inteligentes". Por ejemplo, no llegan a agotar la valiosa memoria del usuario y pueden ejecutar una compleja operación en disco mientras el ordenador prosigue con el programa aplicativo.

En tercer lugar, el DOS puede residir dentro de la RAM del ordenador. Esta técnica se está popularizando cada vez más en los sistemas de gestión empresarial, en los cuales las unidades de disco están incorporadas en el ordenador y la cantidad de RAM disponible es enorme (digamos que el estándar es de más de 128 Kbytes). Desde el punto de vista del fabricante, esto tiene la ventaja de que elimina la necesidad de crear un juego de unidades ROM completamente nuevo cada vez que se modifique el DOS aunque sea en una mínima parte, y el usuario se beneficia al poder escoger un sistema operativo entre los muchos que están patentados y que funcionan con el mismo hardware.

Pero ¿cómo se introduce el DOS en la RAM en primer lugar? Esta pregunta surge de inmediato cuando se enciende el sistema. El DOS ha de transferirse del disco a la RAM, pero si en el ordenador no hay DOS que le diga cómo controlar el disco, ¿cómo podrá cargar algo en la RAM? Un programa no se puede introducir por sus propios medios dentro de la RAM, de modo que el ordenador ha de contar con un pequeño programa incorporado en la ROM para poder ejecutarlo cada vez que la máquina se encienda. Este programa se denomina *bootstrap* y es, en sí mismo, una forma muy sencilla de DOS. La función del bootstrap consiste simplemente en hallar el DOS principal en el disco y transferirlo byte por byte a la RAM, donde ese DOS podrá tomar a su cargo y llevar a cabo algunas



El disco
La superficie de grabación magnética de un disco está dividida en pistas concéntricas y sectores radiales. La intersección de una pista con un sector se denomina bloque, y en él se almacenan, por lo general, 128 bytes. La función del DOS, con la ayuda de un directorio y de un mapa de disponibilidad de bloques (BAM), consiste en controlar lo que hay almacenado en cada bloque.

funciones mucho más sofisticadas. Este proceso de encender el ordenador y esperar después a que el DOS se haga cargo, se denomina *booting up*. Una vez concluido, se imprime un saludo en la pantalla junto con un aviso que indica que el ordenador está preparado para recibir una orden del usuario.

Sea cual fuere la forma que asuma el DOS en un sistema, su principal función es la de buscar las localizaciones del contenido del disco. El lector recordará que un disco (véase p. 114) se divide en ocho anillos concéntricos, denominados pistas, que a su vez se dividen en sectores; y la intersección de una pista con un sector se denomina bloque. Normalmente un bloque retiene hasta 128 bytes de información y constituye la unidad más pequeña que el disco puede leer o escribir a la vez. Para permitirle al ordenador recordar la localización exacta de todo cuanto contiene el disco es indispensable disponer de un DOS. Esta tarea es mucho más abrumadora de lo que parece a primera vista. Supongamos que nuestra unidad de disco posee una capacidad de 320 Kbytes, suficiente para almacenar 20 programas de 16 Kbytes cada uno. Reteniendo cada bloque 128 bytes, para cargar uno de estos programas sin la ayuda de un DOS, el usuario tendría que especificar 128 bloques diferentes, ¡cada uno de ellos con su número de pista y de sector!

Para poder cumplir esta función, el DOS lleva un directorio del disco. Éste, para facilitar su lectura, suele estar situado en la pista central del disco, de esa manera se reduce la distancia a la cual ha de moverse la cabeza de lectura-escritura. La velocidad de funcionamiento de un disco depende más de lo que tarda la cabeza en desplazarse de una pista a otra que de la velocidad a que gira el disco.

El directorio

Nombre del archivo	Tipo	Localización (pista-sector)
Invasores	Progr.	20-1,20-7,20-2...
Temper.	Progr.	25-11,26-5,26-12...
Presup.	Progr.	23-12,24-3,24-9...
Datpresup.	Datos	27-1,27-7,27-2...

El directorio de un disco normalmente ocupa la pista central. Contiene una lista de los nombres de los archivos, su tipo (programación, datos u otras categorías) y los números de la pista y del sector donde se encuentra almacenado el archivo

Se denomina *directorio* a la lista de todos los archivos (que pueden ser de programas o de datos) existentes en el disco; en él se detalla el nombre del archivo, a qué tipo corresponde, y se proporciona una relación de los bloques (cada uno de ellos especificado mediante pista y sector) donde se encuentra almacenado. Puede haber algunas otras entradas, como la fecha en que se hizo la última copia del archivo o una lista de los usuarios que pueden tener acceso a un archivo determinado.

Cuando se ha de almacenar un archivo nuevo, el DOS debe primero consultar lo que se conoce como lista de sectores libres o mapa de disponibilidad de bloques (*Block Availability Map*: BAM). Éste posee un único bit correspondiente a cada uno de los bloques del disco, y cuando se emplea un bloque el valor de su bit se cambia de cero a uno. Algunos ordenadores personales con unidades de disco incorporan un programa de utilidad que visualiza el BAM en la pantalla y el usuario puede ver las entradas que se van efectuando mientras guarda un programa. Cuando se borra un archivo, el DOS no necesita limpiar todos los bloques que se utilizaban para ese archivo; simplemente, cambia las entradas en el BAM para indicar que el contenido de aquellos bloques ahora ya no interesa.

Este sistema tiene, además, otra característica: los archivos no se almacenan, como sería de esperar, en bloques colindantes consecutivos. Supongamos, por ejemplo, que una pista consta de 12 sectores, numerados del 1 al 12 en el sentido de las agujas del reloj. Pues bien, los primeros 128 bytes de un programa podrán hallarse en el sector 1, los segundos en el sector 7, los terceros en el sector 2, y así sucesivamente. Ello se debe a que transcurre un breve lapso mientras el contenido de un bloque se transfiere al buffer de memoria que se utiliza para escribir cada bloque. Si el DOS tuviera que escribir sectores consecutivos, entre cada escritura debería esperar una revolución completa del disco, con lo cual el sistema se retardaría. Además, un disco que haya estado en uso durante algún tiempo, con archivos cuya longitud cambiará día a día, acabará por tener un BAM parecido a un trozo de queso de Gruyère, y los archivos nuevos habrán de acomodarse en los agujeros.

Un sistema operativo de disco posee muchas otras funciones, incluyendo dar formato a discos nuevos (marcando las pistas y los sectores en un disco en blanco y creando un directorio vacío), hacer copias *black-up* y "ordenar" los discos llenos. Versiones más sofisticadas incluyen estructuras para manipulación de datos (véase p. 204).

El espacio a ocupar



Antes de que el DOS pueda almacenar un archivo nuevo y hacer una entrada en el directorio, ha de consultar la "lista de sectores libres" o "mapa de disponibilidad de bloques" (BAM). Ésta es una sección de la memoria en la que cada bit corresponde a un bloque del disco. Un 1 binario indica que el bloque está ocupado, un 0 que está libre (en la ilustración aparecen como cuadrados llenos o vacíos). Las pistas más interiores (las de la parte inferior) poseen menos sectores porque son más cortas



Chris Stevens

Unidad "inteligente"
Algunas unidades de disco contienen su propio microprocesador y su propia RAM. Se dice que estas unidades son "inteligentes", y el DOS está incorporado en forma de ROM. Cuando se utilizan unidades "no inteligentes", el DOS está almacenado dentro del ordenador

Límites exteriores

Aplicándole los accesorios apropiados, el ZX81 se puede ampliar y convertirse en una máquina sofisticada

El ZX81 de Sinclair es, de todos los microordenadores que existen actualmente en el mercado, el que ofrece la mejor relación entre calidad y precio, aun en su forma básica. Pero se encuentran a la venta una cantidad sorprendente de accesorios que lo pueden convertir en un sistema de microordenador notablemente sofisticado. Estas unidades incluyen gráficos en color de alta resolución, síntesis de voz y medios para comunicaciones. Por supuesto, el ordenador en sí mismo tiene algunas limitaciones, pero éstas se pueden remediar con la adición de numerosos artículos de fácil adquisición, como teclados estándar profesionales, memorias de acceso directo (RAM) extras y controladores programables de palancas de mando.

Paquete de RAM
En su forma estándar, el ZX81 sólo posee un Kbyte de RAM, del cual 123 bytes están reservados para las variables del sistema. Por consiguiente, la ampliación de memoria tal vez sea la primera exigencia del nuevo propietario. La ampliación de memoria de Sinclair viene en una sola forma (16 Kbytes), pero otras alternativas ofrecen hasta 64 Kbytes, como la versión Cheetah que muestra la ilustración

Acopladores acústicos
Los moduladores-demoduladores vienen en dos formas: modems de conexión directa, que requieren enchufar un conector adicional en el sistema telefónico, y acopladores acústicos como el Micro-Myte 60, que vemos en la fotografía, que utiliza el propio teléfono directamente.
Los modems de conexión directa, que suelen ser más caros, generan y reconocen señales electrónicas que representan los unos y los ceros de la información que se está recibiendo o transmitiendo. Los acopladores acústicos, que pueden funcionar a pila, traducen los ceros y los unos en tonos audibles para su transmisión a través de la red telefónica, y para recibir la información llevan a cabo el mismo proceso pero a la inversa

ROM en Forth
Los microordenadores ZX de Sinclair utilizan una versión de BASIC algo particular y, aunque no es posible instalar otra diferente, el usuario puede cambiar el lenguaje por completo: a FORTH, por ejemplo. Esto se puede hacer de dos formas: cargando el nuevo lenguaje en la RAM desde una cassette, lo cual significa que el ordenador revertirá al BASIC cada vez que se lo encienda o se lo restaure; o sustituyendo la ROM en BASIC por otra. Esta ROM en FORTH de la firma David Husband llega aún más lejos que la mayoría: permite ejecutar simultáneamente en el ordenador diez o más programas. Sólo es posible obtener total provecho de esta configuración en las aplicaciones de control, donde se deben programar de manera independiente diversos dispositivos

También disponibles...
Además de las unidades que vemos aquí, existen otros dispositivos para mejorar el rendimiento del ZX81. Una tarjeta de colores, por ejemplo, proporcionará hasta 16 colores en la visualización en televisión, y un generador de sonido dará tres "voces" programables. Las puertas bidireccionales pueden admitir hasta 16 dispositivos de entrada/salida a la vez. Lejos de ser sólo un ordenador personal pequeño y nada sofisticado, ideal para jugar y aprender las primeras nociones de la programación BASIC, el ZX81 de Sinclair se puede ampliar para aprovechar al máximo el potencial de que dispone su microprocesador Z80

Marcus Wilson-Smith



Síntesis de voz
Otro interesante accesorio de la firma Cheetah es la unidad para síntesis de voz Sweet Talker. Utiliza un sistema alofónico y, en consecuencia, es mucho más difícil de programar que las unidades que trabajan con fonemas (los alófonos son grupos de fonemas de sonido similar). Existen unidades semejantes para el ZX81 y otros microordenadores personales

Hebot
La tortuga Hebot, que se vende ya montada o en forma de kit, es uno de los más sofisticados robots móviles. Viene con el software para activarla y existe una variedad de extras disponibles, como fotosensores, que se pueden utilizar en unión de una cinta reflectante adherida al suelo, para hacer que el robot siga un camino predeterminado

Teclados
El teclado de membrana de capas múltiples tal vez sea la configuración menos satisfactoria del ZX81, de manera que no es sorprendente que varias empresas ofrezcan teclados alternativos de tamaño natural con teclas de resorte convencionales. El teclado Mapsoft ZX81, que vemos aquí, se vende montado o como kit de montaje. Además del juego de caracteres normal, el teclado Mapsoft proporciona tres teclas extras. Otra posibilidad es un teclado adhesivo de las mismas dimensiones que el propio teclado del ZX81, que se utiliza conjuntamente con el original. Localizar una tecla determinada por medio de él resulta bastante sencillo, pero no tiene ninguna otra aplicación

Mando de palanca
Si se considera que muchas de las unidades ZX81 que se han vendido se utilizan para juegos, quizá resulte extraño que Sinclair no haya fabricado su propio mando de palanca. Sin embargo, existe una gran variedad de ellos en el mercado; pueden ser no programables (especifican por usted los golpes de tecla que efectuará la palanca de mando) o programables (el usuario decide qué teclas se simularán). El modelo que mostramos aquí, el AGF Hardware, se programa moviendo los cables conectores. Otros se programan a través del ordenador. Éste acepta una palanca de mando con cualquier tipo de interruptor, así como una bola de mando

Impresora ZX
La ZX Printer, la impresora Sinclair, utiliza papel aluminizado, que es sensible a la electricidad. En vez de imprimir de una forma convencional, la cabeza de impresión elimina el revestimiento de aluminio para dejar al descubierto la superficie más oscura que se halla debajo. De funcionamiento razonablemente rápido, los principales problemas los constituyen el tipo y la anchura limitados del papel. No obstante, se puede utilizar una impresora normal por medio de una tarjeta para interface. Existen fichas que admiten interfaces Centronics y RS232

A toda marcha

Prestando una cuidadosa atención a las variables y a la estructura del programa, se puede acelerar la operación de prácticamente cualquier programa en BASIC

El BASIC es, a pesar de lo que afirmen sus detractores, un lenguaje versátil y un eficaz auxiliar educativo. Se puede escribir cualquier programa en BASIC siempre y cuando la máquina utilizada posea suficiente memoria y que el tiempo de ejecución no sea importante. Sin embargo, puesto que por lo general el BASIC se interpreta en vez de compilarse (véase p. 184), puede ser extraordinariamente lento al ejecutar los programas, en especial aquellos que exigen traducir y ejecutar repetidas veces una misma instrucción.

La clasificación, por ejemplo, es un proceso sumamente reiterativo: el procedimiento se efectúa dentro de un bucle y hay bucles más pequeños anidados dentro del bucle principal (véase p. 286). Si se han de clasificar 100 ítems, el programa puede efectuar entre 2 500 y 5 000 iteraciones del bucle. Una clasificación en BASIC siempre será lenta, pero la forma en que se escriba el código puede significar una sustancial diferencia en cuanto a la velocidad de ejecución. Si una instrucción se ha de repetir 5 000 veces, y si codificarla adecuadamente puede ahorrar una centésima de segundo del tiempo de ejecución para cada repetición, entonces el ahorro total será de 50 segundos: un considerable progreso para el usuario.

Para apreciar la diferencia que existe entre una buena y una mala codificación, necesitará un mecanismo de tiempos y un programa *testbed* (de apoyo). Si posee un ordenador Commodore, puede utilizar el reloj del sistema, con las variables correspondientes TI$ y TI, como parte del programa *testbed*. Si su ordenador no posee un reloj accesible, habrá de utilizar un cronómetro para medir el código en ejecución. También es una buena idea hacer que su programa le emita un "beep" cuando empiece y cuando termine, para que pueda saber cuándo está funcionando.

El programa *testbed* sería así:

```
1000 L = 500
2000 PRINT "***ADELANTE***":REM "BEEP"
       aquí instrucciones
2100 TI$ = "000000"
2200 FOR K = 1 TO L
..................................................
2900 NEXT K:T9 = TI
2950 REM "BEEP" aquí instrucciones
3000 PRINT "******STOP******"
3100 PRINT "Esto tardó   ";(T9/60);"  segundos"
```

Las líneas 2100 y 3100 son para los usuarios del Commodore. En otros ordenadores, elimínelas o sustitúyalas por el código adecuado. El código a medir lo colocaremos entre las líneas 2200 y 2900. Observe que los cronometrajes se ceñirán a las repeticiones de L donde L es el límite del bucle. Comprobar una sola ejecución de un trozo del código será muy inexacto, porque el reloj del sistema mide sólo en sesentavos de segundos y existe asimismo un tiempo general impuesto por el código del programa *testbed*.

He aquí algunas reglas generales para escribir en un BASIC eficaz, transcritas, aproximadamente, en orden de importancia:

1. Evitar en los bucles todo lo que sea aritmética.

Las funciones exponenciales (x^3, por ejemplo, que significa "x elevado al cubo") y matemáticas (cos(y), por ejemplo, que significa "el coseno del ángulo y") son particularmente lentas. La multiplicación y la división son procesos más lentos que la suma y la resta, pero aun la más rápida de estas operaciones (la suma) es relativamente lenta.

Inserte estas líneas en el programa *testbed*:

```
900 Z = 1.1
2300 X = Z↑3
```

y ejecútelo. En nuestra máquina, 500 repeticiones tardaron 27,95 segundos. Ahora sustituya la línea 2300 por:

```
2300 X = Z*Z*Z
```

y ejecútelo. Ello ocupó 3,55 segundos: ¡una diferencia sustancial!

Una investigación ulterior revelará el nivel de exponenciación en el cual vale la pena reemplazar la multiplicación repetida por la función exponencial. En nuestro ordenador, este nivel estaba en la potencia 18 (cuando X = Z↑18). Recuerde, no obstante, que para calcular $Z^{2,3}$, por ejemplo, la multiplicación repetida sería inútil, mientras que la función exponencial (↑) funciona para todos los números reales, incluyendo los negativos.

Utilice el programa *testbed* para comprobar cuánto tiempo tardan los otros procesos aritméticos, y compare las alternativas. ¿Es más rápido dividir un número por 2 o multiplicarlo por 0,5, por ejemplo?

2. Utilice variables en vez de constantes numéricas.

Cada vez que en una instrucción en BASIC se produce una constante numérica (7.280, por ejemplo), se invierte tiempo en traducir el número a una forma utilizable. Pruebe con esta línea:

```
2300 X = X+7280
```

En nuestra máquina, ésta tardó 4,63 segundos en ejecutar 500 repeticiones, mientras que:

```
900 C = 7280
2300 X = X+C
```

tardó 2,75 segundos en realizar la misma cantidad de repeticiones.

3. Si debe utilizar la sentencia GOTO, salte hacia adelante en su programa en vez de saltar hacia atrás. Sin embargo, si debe saltar hacia atrás, hágalo hasta el principio del programa en vez de saltar hacia atrás unas pocas líneas.

Lo mismo es válido para GOSUB. Al encontrarse con una instrucción GOTO o GOSUB, el intérprete de BASIC compara el número de la línea fijada con el número de la línea en curso. Si el número fijado es mayor que el número de la línea en curso, el intérprete simplemente busca hacia adelante, línea a línea, hasta hallarlo. Pero si el objetivo es menor que el número en curso, entonces la búsqueda siempre empieza por la primera línea del programa. Esto significa que puede ser más eficaz colocar las subrutinas y las secciones utilizadas más frecuentemente a cualquier extremo de un programa. Agregue 56 líneas REM al comienzo del programa para que tenga una longitud típica, y pruebe con:

```
2300 GOTO 2400
2400 GOTO 2500
2500 GOTO 2900
```

Con esto costó 2,33 segundos realizar 500 repeticiones, mientras que:

```
2300 GOTO 2500
2400 GOTO 2900
2500 GOTO 2400
```

ocupó 4,85 segundos.

4. Inicialice todas las variables en orden de frecuencia de acceso.

El intérprete almacena los nombres de las variables en una tabla de símbolos en el orden en que aparecieron por primera vez en el programa. Cuanto más tarde se produzca una variable en la tabla, más tiempo llevará hallarla y acceder a su contenido. Por la misma razón se debería evitar una variable nueva en un programa cuando pueda recurrir a alguna utilizada previamente por el programa pero que no se esté empleando en ese momento.

Si se utiliza una variable dentro de bucles anidados (hecho común en la clasificación), a esa variable se accede con mucha frecuencia, de modo que inicialícela al principio del programa antes que cualquier otra variable, con un valor ficticio en caso de que fuera necesario, como:

```
1000 L = 500:C = 7280:X = 0:Z = 1,1
2300 A = 0
```

que ocupó 2,2 segundos en efectuar 500 repeticiones, mientras que:

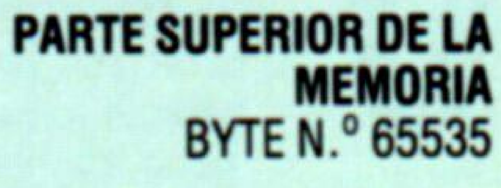
```
1000 A = 0:L = 500:C = 7280:X = 0:Z = 1,1
2300 A = 0
```

tardó 2,06 segundos.

5. Evitar la utilización de series.

Las operaciones en serie consumen mucha más memoria que las operaciones aritméticas, y de vez en cuando podría ser necesario que el intérprete tuviera que llamar a un programa del sistema denominado "recolector de información inservible" para poner en orden los contenidos de la memoria en serie. Este procedimiento puede tardar muchísimo tiempo.

Es difícil escribir una demostración general de ello, debido a la gran variación que experimentan



Mapa de memoria

Éste es el mapa de memoria simplificado de un ordenador personal típico. La mayoría de los microprocesadores pueden direccionar hasta 64 Kbytes (65536 bytes), que se dividirán en la ROM, la RAM y el espacio no utilizado. Al considerar la velocidad de un programa en BASIC, uno de los factores más importantes es la forma en que se almacenan las series. Cada vez que se modifica el contenido de una serie, en la memoria se realizará una copia completamente nueva de la misma. Finalmente, se consumirá toda la memoria libre y el BASIC tendrá que llamar al "recolector de información inservible", que pondrá en orden la memoria en serie. Esto puede durar varios segundos y esto podría significar un considerable retraso para un programa que manipule muchas series

los ordenadores en cuanto a la administración de su memoria: se ha de llenar con datos gran parte de la memoria para el usuario (bastará una gran matriz numérica) y luego efectuar manipulaciones en serie que hagan necesario llamar al "recolector de información inservible". En nuestra máquina dimos entrada a:

```
40 POKE 52,32:POKE 56,32:CLR
```

para reducir severamente la cantidad de memoria disponible para los programas en BASIC, y después dimos entrada a:

```
1000 L = 500:DIMT$(L)
1100 FOR K = 1 TO L
1200 T$(K) = "A"+"B"
1300 PRINT K
1400 NEXT K
```

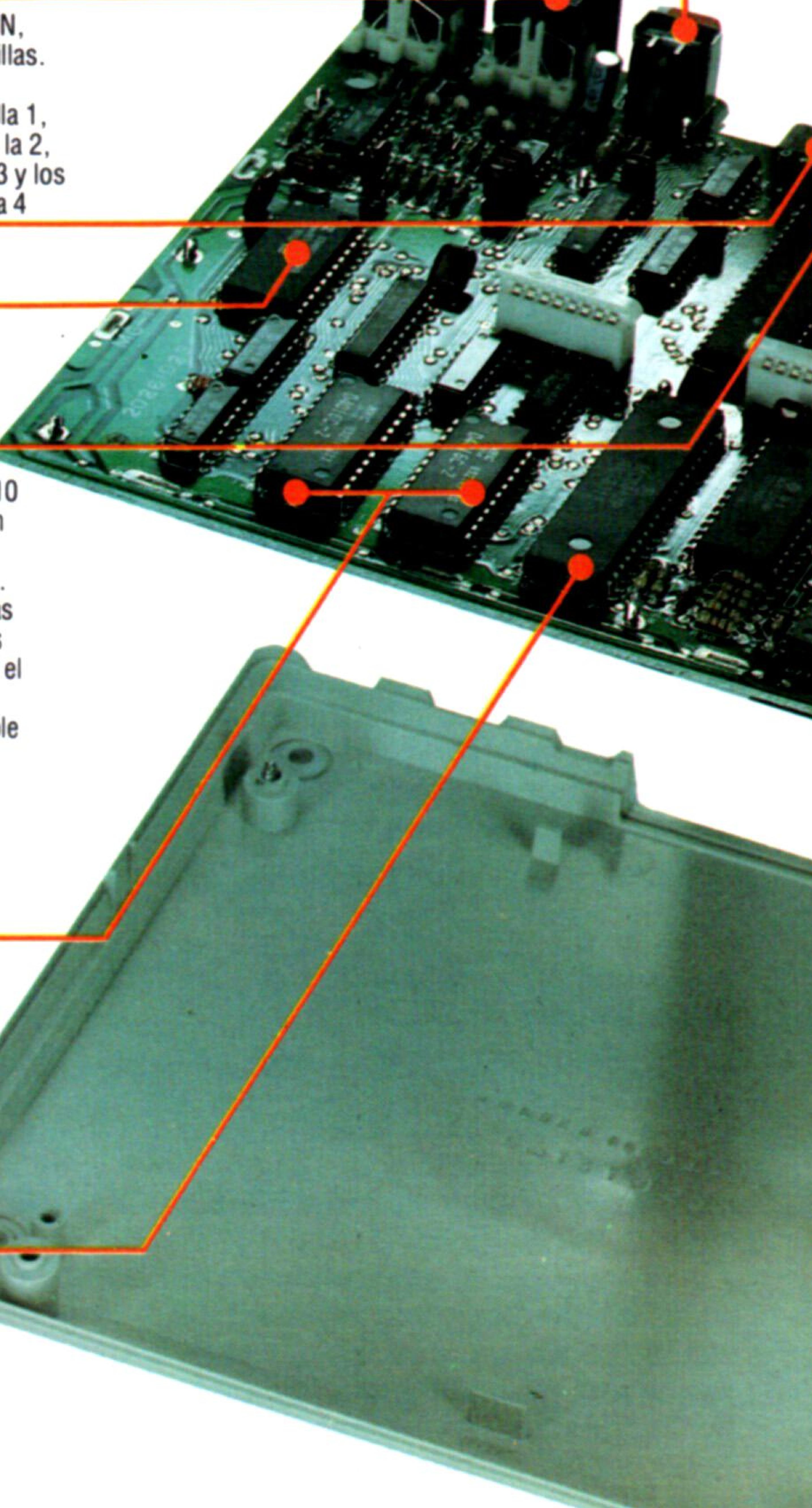
que consume muchísima memoria en serie y proporciona una matriz en serie para su utilización ulterior. La sentencia PRINT se ejecuta en cada iteración, visualizando el valor del contador del bucle. Cuando ejecutamos esta versión del programa *testbed*, la impresión se interrumpía repetidamente mientras se llamaba al "recolector de información inservible" para que reorganizara la memoria. Algunas veces la pausa duró más de tres segundos. El programa continúa:

```
2300 A$=LEFT$(T$(L),1):B$=A$+RIGHT$(T$(L),1)
```

y esto tardó 30,03 segundos en efectuar 500 repeticiones. Cuando ejecutamos el mismo programa con mucha más disponibilidad de memoria, el recolector de información inservible no se vio, y el bucle cronometrado duró 8,66 segundos.

Tandy MC-10

Pese a su precio económico, este ordenador ofrece buen color y configuraciones propias de máquinas más caras



El teclado del Tandy MC-10
El teclado es de los de botón, pero es mejor que muchos otros. Las teclas son de plástico duro, con inscripciones grabadas, que tardan más tiempo en gastarse, y cuenta con una adecuada barra espaciadora. Lamentablemente hay una sola tecla SHIFT, situada a la derecha, y el gran botón de la izquierda corresponde a la tecla de CONTROL, localizada más convenientemente. La sensación táctil de las teclas es cómoda, pero no son aptas para la escritura rápida

Mando de borrado
Por ser grande y de color rojo, el mando de borrado es más fácil de hallar que los de otras máquinas. Al utilizar el MC-10, cuide de no golpear la parte posterior de la máquina en este lugar

Interface para cassette
Las patillas 1 y 3 de este enchufe DIN de cinco patillas proporcionan control remoto. La entrada de señales está en la patilla 4, la salida en la 5 y la señal a tierra en la patilla 2

Interface RS232
Es también un enchufe DIN, pero consta sólo de 4 patillas. La detección de señales portadoras está en la patilla 1, los datos de recepción en la 2, los de señal a tierra en la 3 y los datos de transmisión en la 4

Bus del sistema
No está definido en el manual, pero es obvio que está destinado a ser utilizado con algunas unidades de ampliación aún no especificadas. Hay, no obstante, líneas suficientes para manipular algunos dispositivos complejos

ROM
Está soldada firmemente al tablero y por ello no es probable que se pueda reemplazar por otras versiones. El BASIC Microsoft está almacenado en los 8 Kbytes de ROM

CPU
Insólitamente, el Tandy MC-10 utiliza un procesador 6803 en lugar de uno de los tipos de procesadores más populares. Éste es miembro de una de las familias más antiguas y no es tan conocido como el 6502 o el Z80. Sin embargo, es un útil chip de 8 bits con un razonable conjunto de instrucciones

RAM estática
Los 4 Kbytes nominales de la RAM del usuario están retenidos en estos dos chips de RAM estática de 21 Kbytes × 8 bits, así como la RAM de pantalla y algunas variables del sistema

6847 VDP
Al igual que muchas otras máquinas, la pantalla se controla mediante un chip especial, que en este caso es el 6847 Video Display Processor (procesador de visualización en video). Este chip (al menos en teoría) puede programarse para formatos de pantalla diferentes. No obstante, en la práctica esto se hace en muy raras ocasiones

El Tandy MC-10 es una máquina pequeña y compacta que consigue mucho utilizando unos pocos y sofisticados chips. El teclado es del tipo botón, si bien ligeramente más grande que otros de esa misma clase, y posee una barra espaciadora adecuada. Otras configuraciones hacen que la máquina resulte bastante fácil de utilizar. La entrada de palabras clave en BASIC mediante una única tecla, por ejemplo, se consigue manteniendo pulsada la tecla CONTROL mientras se pulsa la tecla de la función deseada. La máquina se coloca en modalidad "de letras mayúsculas" al encenderse, y la modalidad de letras minúsculas se activa pulsando SHIFT 0 y se desactiva volviendo a pulsar las mismas teclas.

La visualización en pantalla es más pequeña que la de la mayoría de los otros ordenadores personales. Existen sólo 16 líneas de 32 caracteres y sólo se pueden obtener gráficos de una resolución bastante baja. La visualización adolece, asimismo, de otros defectos, incluyendo unas posibilidades de color más bien limitadas, si bien la calidad del color es muy alta. Lo más sorprendente de todo es que no visualiza caracteres en minúsculas, que sí se reconocen pero que en cambio se muestran como letras en mayúscula. El texto sólo puede ser verde sobre negro o viceversa: aunque los símbolos del bloque de gráficos se hallen en la modalidad de cualquiera de los nueve colores disponibles, la letra o el fondo



Disipador
El transistor regulador de potencia Triac se calienta mucho al estar en funcionamiento y esta gran pieza de metal disipa ese calor

Modulador de TV
Convierte el flujo de datos producido por el sistema de circuitos de video en una señal de TV de canal 36, pero sin sonido en la señal de TV. Ésta es la única salida de pantalla y la máquina no dispone de conector para monitor

Enchufe red
Se trata de un conector coaxial normal de poco voltaje. Al igual que otras máquinas de este tipo, el Tandy MC-10 toma su energía de un pequeño transformador de poco voltaje conectado a un enchufe

Regulador de potencia
Este gran transistor, junto con otros componentes cercanos a él, estabiliza la potencia transformada pero no regulada

Interruptor de potencia
Dado que el MC-10 posee un mando de borrado, no es necesario utilizarlo como alternativa, como ocurre con otras máquinas

Cristal
El reloj maestro genera una frecuencia de 4,4 MHz; está subdividido en impulsos más lentos y se lo utiliza para toda la máquina

Chris Stevens

TANDY MC-10

DIMENSIONES
210 ×178 ×51 mm

CPU
6803

VELOCIDAD DEL RELOJ
4,4 MHz

MEMORIA
8 Kbytes de ROM
4 Kbytes de RAM

VISUALIZACION EN VIDEO
16 líneas de 32 caracteres, 9 colores, pudiéndose determinar sólo el fondo. 75 caracteres predefinidos

INTERFACES
Interface RS232 en serie, cassette

LENGUAJE SUMINISTRADO
BASIC

OTROS LENGUAJES DISPONIBLES
Ninguno

VIENE CON
Manual de funcionamiento y manual de BASIC, cable para TV

TECLADO
48 teclas tipo botón

DOCUMENTACION
Clara, adecuada y bien diseñada, pero algo carente de información técnica. El único fallo importante es la ausencia de índice. Se incluye una ficha de referencia rápida, que da del BASIC suficientes detalles para que una persona experimentada pueda empezar a trabajar con la máquina enseguida

han de ser siempre negros. Por consiguiente, no se puede producir una forma azul sobre un fondo rojo, ¡ni siquiera en la modalidad de gráficos!

La función de sonido también tiene limitaciones. Sólo hay un canal disponible, que permite solamente variaciones mínimas de altura y de duración. Las facilidades de input/output son para cassette (incluyendo control remoto), televisión y una puerta en serie RS232. La puerta en serie se puede utilizar como línea para transferir datos hacia y desde otros ordenadores o, alternativamente, para activar una impresora. También se puede utilizar para crear una red con otros ordenadores Tandy MC-10.

Los diseñadores de la máquina no parecen haberse ocupado de manera especial de los juegos, ya que no han dotado al ordenador de conexiones para palanca de mando o mando de raqueta ni de ninguno de los chips especiales controladores de sonido y de gráficos existentes en otras máquinas más idóneas para juegos.

No obstante, para el futuro están claramente perfiladas algunas posibilidades de ampliación, dado que en un conector marginal hay una terminación de bus de sistema muy misteriosa, tapada con una placa atornillada. Aparte de afirmar que “esta ranura está reservada para futuras unidades de ampliación de memoria”, el manual no dice nada más acerca de ella y no proporciona ninguna pista en cuanto a cuáles serán los accesorios que se podrán enchufar en tal ranura.

La documentación del Tandy MC-10 es la clásica documentación de las otras máquinas Tandy: texto bastante sólido, con pocas imprecisiones.

Por su precio asequible, vale la pena tener en cuenta este ordenador, pero al leer las especificaciones recuerde que por más que posea 4 Kbytes nominales de RAM, sólo son 3,142 bytes los que están disponibles para el usuario, porque de allí se toman también la RAM de pantalla y algunas variables del sistema.

Nuevos mandos

Dos innovadores tipos de palanca de mando. Una utiliza interruptores de mercurio; la otra capta las señales electromagnéticas del cuerpo humano

La industria del ordenador personal se ha habituado a un desarrollo tecnológico rápido, y los cambios no se limitan a los ordenadores: los periféricos y los accesorios están también sujetos a constantes refinamientos. Por ejemplo, en el breve tiempo transcurrido desde que analizáramos por primera vez el mecanismo de una palanca de mando (véase p. 56), han aparecido en el mercado dos tipos completamente nuevos de estos dispositivos.

Un dispositivo denominado *Le Stik* ha sido la primera palanca de mando analógica que ha dejado de lado los mecanismos de señalización habituales. Consiste en una empuñadura contorneada, provista de un pulsador de disparo montado en la parte superior y un control de pausa dispuesto en uno de los lados. La palanca de mando se sostiene en el aire y se inclina respecto a la vertical en la dirección requerida, y en la pantalla la imagen correspondiente se mueve de acuerdo con ella.



Manos arriba
La Trickstick se basa en el "zumbido" de la red, que es la radiación electromagnética que emite en todas las casas la red eléctrica. El cuerpo humano actúa como si fuera la antena de este "zumbido", y los sensores de la palanca captan los diferentes niveles de "zumbido" de acuerdo con la presión que se aplique con los dedos

Trickstick

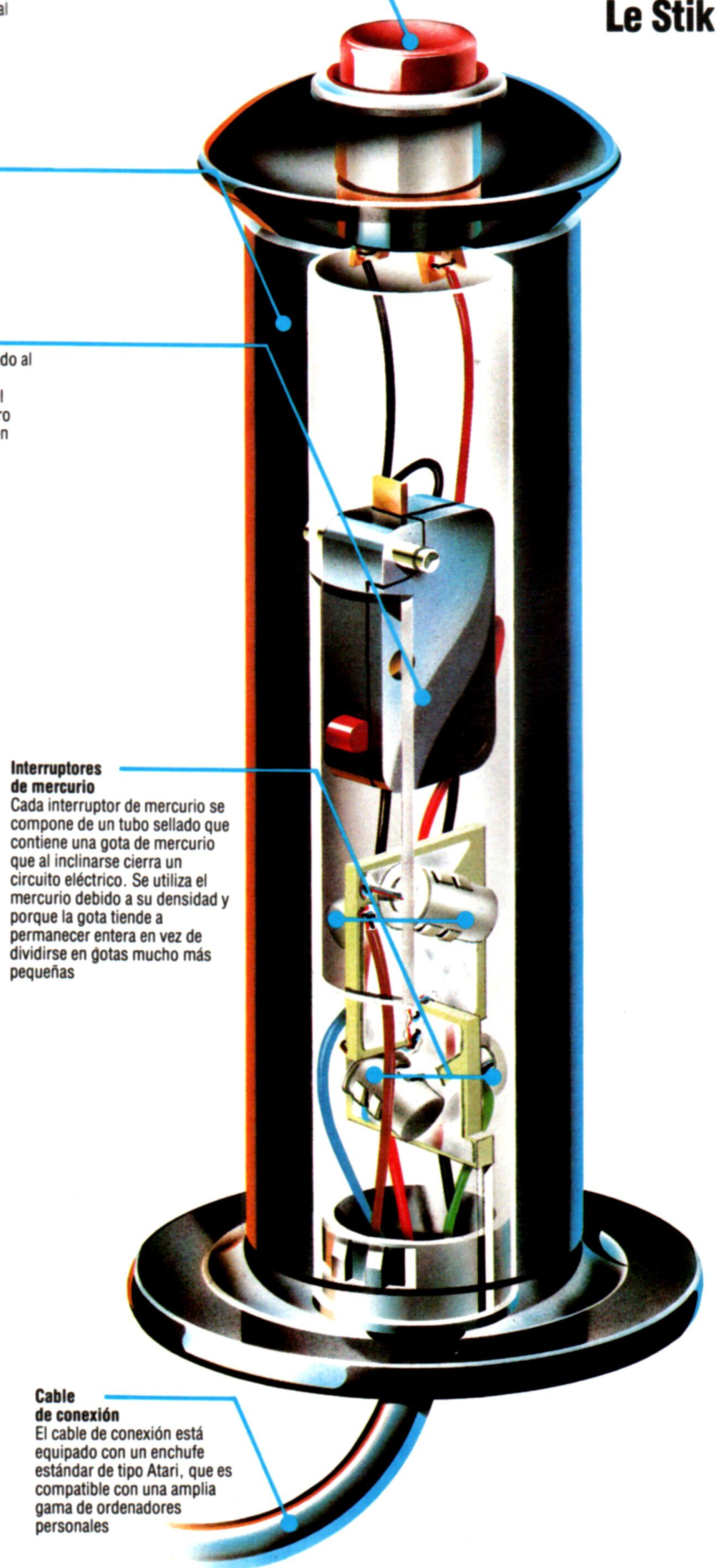
Controles de movimiento horizontal
Balanceando el pulgar sobre estos dos sensores se puede controlar el movimiento hacia atrás y hacia adelante

Controles de movimiento vertical
El sensor superior controla el movimiento hacia arriba; el inferior controla el movimiento hacia abajo

Control de sensibilidad
Permite ajustar la Trickstick al nivel de sensibilidad "emitida" por el usuario

Pulsadores de disparo
Cada uno genera una señal independiente, de modo que con un pulsador se podrían arrojar bombas mientras con el otro se disparan rayos láser

Le Stik



El mecanismo esencial de *Le Stik* se compone de cuatro tubos sellados llenos de mercurio. A medida que la palanca de mando se inclina respecto a la vertical, el mercurio fluye en la dirección elegida y hace uno o varios contactos eléctricos, como si se hubiera cerrado un interruptor. Al mover la empuñadura para que recupere la posición vertical, el mercurio vuelve a fluir en los tubos, cesando, de este modo, el contacto. La respuesta del sistema es mejor que la de las palancas de mando anteriores.

Por su parte, la *Trickstick*, diseñada por la East London Robotics', utiliza el más reciente procedimiento para transformar los movimientos manuales en señales que pueda comprender un ordenador. En efecto, esta palanca de mando constituye algo único en cuanto al efecto eléctrico que emplea: utiliza al cuerpo humano como una antena para captar el "zumbido" de la red eléctrica (la radiación electromagnética inocua que emite en cualquier habitación la red eléctrica). La *Trickstick* consiste en un tubo sellado dentro de una carcasa plástica, que se sostiene verticalmente entre ambas manos. En la superficie del tubo hay tres pares de sensores: un par controla el movimiento hacia adelante y hacia atrás; otro par controla el movimiento hacia arriba y hacia abajo y el par restante corresponde a los pulsadores de disparo.

El "zumbido" que capta el cuerpo humano se transmite a través de estos sensores al sistema de circuitos sensible, donde los impulsos se convierten en señales que le proporcionan al ordenador información direccional. Las señales también se pueden analizar para mostrar hasta dónde ha de llegar el movimiento. Cuanto más fuerte se pulse, más potente será la señal y más rápida la salida al ordenador. De este modo, la *Trickstick* combina el control proporcional de la palanca de mando analógica con el rápido control digital directo de una unidad basada en interruptor. Dado que diferentes personas afectarán a los circuitos de modo diverso, la *Trickstick* se ha de ajustar a la sensibilidad de cada usuario. Esto se realiza por medio de una perilla montada en uno de los extremos del dispositivo.

La idea de la *Trickstick* es ciertamente interesante, y los fabricantes se han apresurado a solicitar patente para esta técnica. No obstante, su fiabilidad y rendimiento aún están por ver.

Sistemas de sonido

Un segundo análisis de la capacidad de sonido del Vic-20

La última vez que analizamos el Vic-20 en esta sección, vimos de qué manera se pueden controlar los tres osciladores de la máquina "empujando" (POKE) las localizaciones de memoria, cómo establecer los niveles de volumen y cómo controlar la duración de una nota. Investigamos cómo se podían determinar la duración de las notas y las pausas entre ellas mediante la utilización de bucles FOR...NEXT o, con mayor eficacia, a través del empleo del reloj del Vic-20 para contar en *jiffys* (sesentavos de segundo). El manejo de estos tres elementos musicales (frecuencia, volumen y tiempo) le permite al usuario crear melodías sencillas en el Vic-20 y producir efectos sonoros.

Tocando melodías

Para crear una melodía, primero debe ensamblar las notas requeridas. Éstas podrían ser, por ejemplo, las notas de la primera línea de *Oh, I do like to be beside the seaside* ("Oh, quiero estar a la orilla del mar"). Por el orden correcto, éstas se podrían seleccionar como:

Re# Mi Fa Re# Do La# Sol# Sol Sol# Re# Re#

Utilizando las técnicas explicadas en la página 284, la duración de las notas y las pausas se podría establecer utilizando la configuración TI. Nuestra melodía, por lo tanto, se puede tocar ejecutando (RUN) el siguiente programa (observe la utilización de las variables para simplificar la selección de órdenes POKE):

```
10 V = 36878
20 FOR I = 1 TO 11
30 READ N: REM *NOTA*
40 POKE V,7:P = TI: REM *VOL ON*
50 IF TI−P<15 THEN 50: REM *PAUSA*
60 POKE V−3,N:D = TI: REM *TOCAR NOTA*
70 IF TI−D<20 THEN 70: REM *DURACION*
80 POKE V−3,0: REM *STOP NOTA*
90 NEXT I
100 DATA 203,207,209,203: REM *VALORES
    NOTAS*
110 DATA 195,187,179,175
120 DATA 179,203,203
130 POKE V,0: REM *VOL OFF*
140 END
```

Programa de luz

Primeros pasos con los sofisticados gráficos del BBC

El BBC Micro puede proporcionar unos efectos gráficos verdaderamente impresionantes tan sólo con unas pocas líneas en BASIC.

En el BASIC del BBC existen varias órdenes de alta resolución, incluyendo instrucciones para trazar líneas rectas, colocar puntos y trazar y rellenar triángulos. Esta última función se emplea para colorear el interior de las formas mediante una serie de triángulos pequeños, ya que este ordenador no dispone de ninguna orden del tipo PAINT (pintar). Carece, asimismo, de orden en BASIC para dibujar círculos y elipses, y carece de configuraciones para la programación de sprites. No obstante, posee características interesantes y poco frecuentes de las que carecen la mayoría de sus competidores. Éstas incluyen la posibilidad de mezclar texto y gráficos en la pantalla, cursores de texto y de gráficos que se pueden controlar por separado, y el acceso a la parte del sistema operativo de la máquina que controla la visualización en pantalla, desde dentro de un programa en BASIC. Esto se consigue mediante el conjunto de órdenes de pantalla o VDU. Las "ventanas" de texto y de gráficos también se pueden definir en la pantalla, lo que le permite al usuario dividir la visualización en sectores separados para gráficos y para texto.

Modalidades de visualización

El BBC Micro posee ocho modalidades para gráficos, tres de las cuales admiten sólo visualizaciones de texto. Se puede escoger entre 20, 40 u 80 caracteres a lo ancho de la pantalla, según la modalidad que se haya seleccionado. Los colores disponibles son dos, cuatro o dieciséis, dependiendo su número, nuevamente, de la modalidad seleccionada; no obstante, esta modalidad de color limitado posee una característica muy interesante: dos o cuatro colores de los que se pueden utilizar no son fijos y pueden ser seleccionados por el programador de entre los 16 normalmente disponibles.

MODE 7 (modalidad 7) se diferencia de todas las demás en que en ella no se emplean el juego estándar de caracteres ASCII y los códigos relacionados con ellos. En cambio, la visualización se construye con caracteres de teletexto. Las órdenes para gráficos normales, como PLOT y DRAW, no funcionan en MODE 7.

La siguiente tabla muestra las opciones de color y de resolución según la modalidad elegida:

Modalidad	Texto	Gráficos	Colores
0	80 ×32	640 ×256	2
1	40 ×32	320 × 256	4
2	20 ×32	160 ×256	16
3	80 ×25		2 (blanco y negro)
4	40 ×32	320 × 256	2
5	20 ×32	160 ×256	4
6	40 ×25		2 (blanco y negro)
7	40 ×25		Teletexto

Este programa simplemente toca las notas en la secuencia correcta con pausas y duraciones iguales. Por consiguiente, la melodía resultante es algo artificial. Si continúa experimentando, podrá crear programas más complejos que proporcionen intervalos y duraciones diferentes para cada una de las notas.

Efectos sonoros

Utilizando dos o tres osciladores se pueden tocar acordes sencillos. El programa que reproducimos abajo ejecuta el acorde en *re mayor* (*fa#*, *la* y *re*), empezando con el *fa#* solo y agregando el *la* y el *re* después de unas demoras establecidas en un segundo cada una. Después el acorde continúa durante dos segundos más.

```
10 POKE 36878,7
20 POKE 36874,233:D = TI
30 IF TI–D<60 THEN 30
40 POKE 36875,219:D = TI
50 IF TI–D<60 THEN 50
60 POKE 36875,147:D = TI
70 IF TI–D<120 THEN 70
80 POKE 36878,0:POKE 36874,0
90 POKE 36875,0:POKE 36876,0
100 END
```

No obstante, existen varios recursos para hacer más atractivo el tono de estos sonidos. Por ejemplo, el volumen se puede variar durante la duración de una nota, haciéndolo subir o bajar de acuerdo a una variable. Por ejemplo:

```
100 V = 36878
110 FOR I = 1 TO 12
120 POKE V,I
130 NEXT I
140 POKE V,0
```

Esto hace que el volumen vaya aumentando gradualmente desde 1 hasta alcanzar a 12 —la escala completa va desde 0 (apagado) hasta 15 (alto)—. Asimismo, el volumen se puede hacer "vibrar" alternando un nivel de volumen alto con uno bajo. La frecuencia se puede variar para "curvar" una nota modificando como sigue la línea 120 anterior:

```
POKE V–3,203+I
```

También vale la pena probar distintas combinaciones de ruido, frecuencias de oscilador y volúmenes. Esto a menudo resulta en un timbre más agradable. Tanto al hacer música como al agregarles efectos sonoros a los juegos, el objetivo de la informática es el de incentivar el interés del usuario, evitando la repetición constante de notas monótonas.

Hemos mostrado cómo se pueden manipular las sencillas configuraciones de sonido del Vic-20 para producir interesantes tonos y secuencias de notas. El principal problema es la falta de órdenes de sonido, lo que obliga a utilizar complicadas sentencias en BASIC para efectuar tareas relativamente sencillas. Esto desemboca en largas rutinas de programa que impiden que el intérprete de BASIC pueda procesar el código entre las notas con suficiente rapidez. La única forma sencilla de evitar este inconveniente consiste en invertir en uno de los muchos paquetes de programas comerciales que proporcionan órdenes extras para la programación de música. El cartucho Super Expander de Commodore proporciona una útil gama de órdenes para sonido, así como un recurso para almacenar melodías escritas con la ayuda del cartucho. No obstante, si usted busca un ordenador que posea configuraciones que le permitan ir más allá de la creación de música o sonidos elementales, sería conveniente que probara otros modelos, como el BBC Micro, el Commodore 64, el Dragon 32 o el Oric-1.

La pantalla de alta resolución se define con su origen en la esquina inferior izquierda de la pantalla, independientemente de la modalidad seleccionada. Los valores verticales oscilan entre 0 y 1023, y los horizontales entre 0 y 1279. Este procedimiento de delinear el mapa de la pantalla resulta muy conveniente cuando el usuario decide cambiar la visualización de una modalidad a otra. Conviene agregar que si en el curso de un programa se cambiara la modalidad de visualización, entonces la pantalla se limpiaría automáticamente.

Los colores del fondo, de los textos y de los gráficos se establecen utilizando las órdenes COLOUR y GCOL. El BBC Micro emplea la interesante idea de los colores lógicos y reales para permitir que el usuario seleccione un juego limitado de colores de entre los 16 disponibles. Para ilustrar esto, lo mejor es utilizar el ejemplo de emplear color en MODE 0, en la que sólo se pueden especificar dos colores. A los dos colores posibles para el primer plano se les dan los números de color lógicos 0 y 1 y, a menos que se instruya al ordenador en otro sentido, éste considera que 0 es negro y 1 blanco. La orden COLOUR selecciona el color de primer plano del texto. COLOUR 1 selecciona el número de color lógico 1 como el color del texto, pero se puede borrar el color de texto lógico utilizando una de las órdenes VDU. VDU19 define el color lógico. Para establecer el color lógico 1 en verde (número de color real 2) se necesita la siguiente orden:

```
VDU19, 1, 2, 0, 0, 0,
```

Los tres ceros a la derecha no tienen valor y están allí para una futura ampliación del sistema.

La orden GCOL tiene dos números relacionados con ella. El segundo es el número de color lógico para la visualización de gráficos; el primero se relaciona con la forma en que ese color se utiliza en la pantalla. Para la orden GCOL a,b los valores de a pueden oscilar entre 0 y 4, permitiéndole al usuario especificar si el punto o la línea se ha de visualizar en el color de primer plano lógico, si se ha de combinar mediante AND, OR o, a través de un OR exclusivo, con el color ya existente, o si se ha de invertir el color original.

En un futuro capítulo volveremos a ocuparnos del BBC Micro y explicaremos sus posibilidades de alta resolución, cómo definir caracteres y analizaremos con mayor profundidad el conjunto de órdenes VDU.

Cambiando de lugar

Tras analizar cómo insertar registros nuevos, seguimos con las formas de recuperarlos. Como ya dijimos, nos encontramos primero con el problema de hallar un emparejamiento exacto

El capítulo anterior finalizó con un ejercicio en el que se debía escribir un programa de tipo base de datos que permitiera dar entrada en ella a información. Analicemos algunos de los pasos que implica dar entrada a un registro nuevo como forma de continuar nuestro estudio de lo que entraña la etapa INICIALIZACION de nuestro programa principal. Daremos por sentado que existen los campos siguientes y las correspondientes matrices:

CAMPO	MATRIZ
1 campo del NOMBRE	NOMCAM$
2 campo del NOMBRE MODIFICADO	MODCAM$
3 campo de la CALLE	CALLECAM$
4 campo de la CIUDAD	CIUCAM$
5 campo de la PROVINCIA	PROVCAM$
6 campo del NUMERO DE TELEFONO	TELCAM$
7 campo del INDICE	INDCAM$

El significado de la mayoría de estos campos debería estar bastante claro, quizá con la excepción de los campos 2 y 7. Consideremos en primer lugar el campo del NOMBRE MODIFICADO. Cuando analizamos por primera vez el problema del formato de los datos para el nombre, dudamos entre dos opciones: darle el formato del nombre estrictamente especificado (rígido) o especificado con flexibilidad (libre), y optamos por esta última. Dado que la forma en que se puede dar entrada a un nombre es sumamente variable, un formato rígido hubiera dificultado en gran medida las rutinas de búsqueda y clasificación. Para obviar este inconveniente decidimos convertir todos los nombres a un formato estandarizado: todas las letras irían de mayúscula, todos los caracteres no alfabéticos (como espacios, puntos, apóstrofos, etc.) se suprimirían y sólo habría un único espacio (en caso de haber alguno) entre el nombre de pila y el apellido.

La necesidad de estandarizar los nombres de esta manera se plantea porque las rutinas de clasificación y de búsqueda han de tener alguna forma de comparar las semejanzas de unos con otros. Por otra parte, cuando recuperamos un nombre y una dirección de la base de datos, queremos que los datos se presenten en la misma forma en que se les dio entrada originalmente. Existen dos maneras de manejar este problema: o cada uno de los nombres archivados se convierte a la forma estándar sólo cuando se están llevando a cabo clasificaciones y búsquedas, o bien el campo del nombre se puede convertir a la forma estándar y almacenar como un campo separado, de modo que las rutinas de clasificación y búsqueda puedan tener acceso instantáneo a los nombres estandarizados.

El otro campo que podría inducir a confusión es el del INDICE. En realidad éste se incluye como un campo separado para permitir la futura ampliación o modificación de la base de datos sin que haya necesidad de reescribir una porción considerable del programa. Su inclusión introduce el tema del *encadenamiento* (término que se refiere a la determinación de las relaciones de los datos y el procesamiento). Todos los campos o elementos de cada uno de los registros están encadenados porque todos poseen el mismo índice (el mismo número de elemento o subíndice en sus respectivas matrices), y porque todos los campos de un registro se almacenarán juntos en un archivo. Esto puede hacer que, en una etapa ulterior, agregar nuevos tipos de datos o de relaciones resulte una tarea difícil que posiblemente implique la reorganización total de la estructura de archivos y la reescritura de gran parte del programa. La incorporación del campo del INDICE en esta etapa simplificará muchísimo la futura incorporación de modificaciones al programa.

Antes de intentar agregar un registro nuevo a la base de datos enunciaremos algunos supuestos relativos a la estructura de los archivos. En primer lugar limitaremos el número de registros a 50. Asimismo, vamos a suponer que ya se han transferido todos los datos a las matrices, como parte del procedimiento INICIALIZACION.

Cuando se añade un registro nuevo, lo más sencillo es agregarlo al final del archivo (es decir, en el primer elemento libre de cada matriz). Existen muchas probabilidades de que el registro nuevo esté desordenado en relación a los otros, pero éste es un problema que podremos investigar más adelante. Lo primero que se habrá de hacer, por lo tanto, será averiguar cuán grande es la matriz. Dado que ésta es una información que probablemente nos será de utilidad en muchas partes del programa, el mejor lugar para efectuarla es en INICIALIZACION. Éste es un caso en el que la necesidad de una variable global es muy clara (es decir, una variable que se pueda emplear en cualquier parte del programa). La denominaremos TAMAÑO. Otra variable global que probablemente tenga utilidad es el índice del registro en curso. Puesto que no habrá ningún registro en curso cuando se ejecute el programa por primera vez, para asignarle a CURSO un valor inicial habrá que esperar hasta que el programa haga algo con los datos. No obstante, CURSO se puede inicializar en 0 en el procedimiento INICIALIZACION. En BASIC no es estrictamente necesario inicializar una variable en cero, porque esto se realiza de forma automática. A pesar de ello es una buena costumbre y se debería hacer siempre con las variables locales para evitar los "efectos colaterales" derivados de la utilización de la misma variable en algún otro lugar del programa.

Cuando se ejecute el programa por primera vez, se producirán varios tipos de inicializaciones y los datos se habrán de cargar desde el disco o la cinta y transferir a variables alfanuméricas. Entonces se presentará el menú ELECCION. Si el usuario escoge

la opción 6 (agregar un registro en el archivo), se devolverá el valor 6 de la variable OPCION, que llamará al subprograma INCLREG. INCLREG supondrá que a TAMAÑO ya se le habrá asignado un valor y que, por lo tanto, puede empezar a solicitar entradas (también supone que INICIALIZACION ha DIMensionado correctamente las matrices necesarias).

Agregar un registro nuevo significa, asimismo, que ahora el archivo está, al menos en potencia, desordenado. Puesto que una clasificación podría ocupar cierto tiempo, tal vez fuera innecesario clasificar los registros después de efectuar cada adición; pero ésta es una decisión que por el momento vamos a posponer. En cambio, estableceremos una bandera para indicar que se ha agregado un registro nuevo.

Ahora estamos en condiciones de empezar a elaborar una lista provisional de las posibles matrices, variables y banderas que se podrían necesitar en el programa.

MATRICES

NOMCAM$	(campo del nombre)
MODCAM$	(campo del nombre modificado)
CALLECAM$	(campo de la calle)
CIUCAM$	(campo de la ciudad)
PROVCAM$	(campo de la provincia)
TELCAM$	(campo del número de teléfono)
INDCAM$	(campo del índice)

VARIABLES

TAMAÑO	(tamaño normal del archivo)
CURSO	(índice del registro en curso)

BANDERAS

RADD	(agregado registro nuevo)
SORT	(clasificado desde modificación del registro)
SAVE	(guardado desde modificación del registro)
RMOD	(efectuada modificación desde que se guardara por última vez)

Es probable que mientras se desarrolle el programa se presente la necesidad de incluir algunas matrices más. Ciertamente, será indispensable utilizar más variables. En cuanto a las banderas, es evidente que si bien se necesitarán otras, puede que no se requieran las cuatro que hemos reseñado arriba. No habrá necesidad ni de guardar ni de clasificar el archivo (se supone que ya está guardado y clasificado) a menos que se haya efectuado una modificación, de modo que quizá la única bandera realmente necesaria sea RMOD. Pero si decidimos emplear las cuatro banderas, el subprograma de INICIALIZACION las habría de establecer a todas en sus valores apropiados. Como ejercicio adicional en cuanto al refinamiento de la programación *top-down*, veamos lo sencillo que es codificar *INCLREG*

I 4 (EJECUCION) 6 (INCLREG)

```
EMPEZAR
  Localizar tamaño habitual del archivo
  Solicitar entradas
  Asignar las entradas a los finales de las matrices
  Establecer bandera RMOD
FIN
```

II 4 (EJECUCION) 6 (INCLREG)

```
EMPEZAR
  (el tamaño del archivo es TAMAÑO)
  (solicitud de entradas)
  Limpiar pantalla
  Imprimir mensaje solicitando la primera matriz (TA-
    MAÑO)
  Dar entrada a los datos para la matriz (TAMAÑO)
  (solicitud y entrada para todas las matrices)
  Establecer RMOD en 1
FIN
```

Todo esto es directo y no incluye bucles ni ninguna otra estructura complicada. El paso siguiente puede ser la codificación directa en BASIC. Los únicos puntos que debemos considerar son que TAMAÑO es una variable que se establece durante la ejecución de INICIALIZACION y que no necesita codificarse como parte de esta sección.

III 4 (EJECUCION) 6 (INCLREG) CODIFICACION EN BASIC

```
CLS: REM O UTILIZAR PRINT CHR$(24) ETC
  PARA LIMPIAR PANTALLA
INPUT "DE ENTRADA AL
  NOMBRE";NOMCAM$(TAMAÑO)
INPUT "DE ENTRADA A LA
  CALLE";CALLECAM$(TAMAÑO)
INPUT "DE ENTRADA A LA
  CIUDAD";CIUCAM$(TAMAÑO)
INPUT "DE ENTRADA A LA
  PROVINCIA";PROVCAM$(TAMAÑO)
INPUT "DE ENTRADA AL NUMERO DE
  TELEFONO";TELCAM$(TAMAÑO)
LET RMOD = 1
LET INDCAM$ = CALLE$(TAMAÑO)
GOSUB *MODNOMBRE*
RETURN
```

La antepenúltima línea establece el campo INDCAM$ en el valor de TAMAÑO (convertido por CALLE$ en una serie), de modo que en una etapa ulterior pueda actuar como un índice. La subrutina *MODNOMBRE*, a la que se llama justo antes de que termine el programa, no es otra que el programa descrito con todo detalle en la página 254. A ese programa será necesario efectuarle algunas ligeras modificaciones, pero estas sólo son detalles. Esta subrutina tiene la función de tomar la entrada corriente (libre) del nombre y convertirla en una forma estándar. La salida de esta subrutina será un elemento (TAMAÑO) de una matriz denominada MODCAM$. Ahora todas las búsquedas y clasificaciones de nombres se pueden dirigir hacia los elementos de MODCAM$ y, dado que el elemento tendrá el mismo índice que los otros campos del registro, será fácil visualizar el nombre y la dirección tal como se les dio entrada originalmente. En otras palabras, la búsqueda se llevará a cabo en MODCAM$, pero la visualización provendrá de NOMCAM$.

Y esto es todo cuanto implica agregar un registro nuevo en el archivo, si bien no hemos dado cabida a ninguna verificación de error, ni hemos tomado medidas para la eventualidad de que no quedara más espacio en la matriz. Puesto que todos nuestros programas los estamos escribiendo en forma modular, las modificaciones y las mejoras de este tipo se podrán efectuar fácilmente después, sin tener que volver a escribir todo el programa.

Los subprogramas MODREG y BORREG (para modificar y borrar registros, respectivamente) son bastante similares a INCLREG, excepto en que antes de que se puedan ejecutar primero debemos localizar el registro que deseamos modificar. Por consiguiente, estos dos subprogramas comenzarán llamando a

Tony Lodge

ENCREG. Este subprograma se basa en una rutina de búsqueda similar a la descrita en la página 273. Esta vez la diferencia principal estriba en que (con toda probabilidad) no habrá ningún dato que sea idéntico a otro, porque muy pocas personas tienen exactamente el mismo nombre.

Una búsqueda se puede llevar de dos formas. Una consiste en buscar en una pila desordenada, lo cual impide que la indagación sea tan expedita como sería de desear. En el peor de los casos, puede que la rutina haya de buscar a través de toda la información antes de localizar el dato buscado. Sin embargo, buscar en una pila desordenada tiene la ventaja de que no se necesitan rutinas de clasificación cada vez que se agrega, se borra o se modifica un registro.

Si la información está ordenada de alguna manera (ya sea numérica o alfabéticamente, por ejemplo) el programa sólo habrá de buscar a través de una pequeña fracción de los datos de la lista. Cuanto más larga sea la lista, tanto más eficaz resultará la búsqueda binaria en comparación con la búsqueda a través de una pila desordenada. En realidad, si en el archivo hay datos suficientes como para garantizarlo, la clasificación de los registros después de una modificación se puede acelerar realizando una búsqueda preliminar para localizar la primera y la última aparición en la matriz de la letra inicial del apellido en el registro en cuestión.

Otra forma de acelerar la rutina de clasificación podría ser mantener una tabla de búsqueda de las localizaciones en la matriz donde conste la primera aparición de cada una de las letras del alfabeto. Sin embargo, esta tabla debería mantenerse (actualizarse) con sumo cuidado cada vez que se produjeran cambios en los datos.

El tema de la búsqueda y la clasificación constituye una de las áreas más extensas de la programación, y a él se han dedicado libros enteros. Nosotros no vamos a intentar descubrir la solución óptima para nuestro programa de agenda de direcciones, porque éste depende de una gran cantidad de factores, incluyendo el número de registros del archivo y también la posibilidad de disponer o no de unidades de disco.

A continuación ofrecemos un programa en seudolenguaje para efectuar una búsqueda a través de los elementos de la matriz MODCAM$. La variable alfanumérica CLY$ es la clave para la búsqueda. En este contexto, la palabra "clave" significa el grupo de caracteres de identificación utilizados para especificar qué registro (o registros) se necesita.

```
Solicitar el nombre a buscar
LET CLV$ = nombre (a buscar)
LET BMT = 1
LET BUSCA = 0
LET TOP = TAMAÑO
LOOP mientras (BTM< = TOP) AND (BUSCA = 0)
  LET MID = INT((BTM+TOP)/2)
  IF CLV$ = MODCAM$(MID)
    THEN
      PRINT NOMCAM$(MID)
      PRINT CALLECAM$(MID)
      PRINT CIUCAM$(MID)
      PRINT PROVCAM$(MID)
      PRINT TELCAM$(MID)
      LET BUSCA = 1
    ELSE
      IF CLV$>MODCAM$(MID)
        THEN LET BTM = MID+1
        ELSE LET TOP = MID-1
      ENDIF
  ENDIF
ENDLOOP
IF BUSCA = 0 THEN PRINT "REGISTRO NO
  HALLADO"
END
```

Este trozo de seudolenguaje se basa mucho en el programa utilizado para buscar los marcadores de fútbol de la página 275 de *Mi Computer*, pero verá que posee una salida adecuada si no se consigue hallar el registro (la última sentencia PRINT), que se ejecutará sólo si el bucle fracasa en localizar un emparejamiento exacto entre CLV$ y MODCAM$(MID).

Lamentablemente, es bastante improbable que se llegue a hallar un emparejamiento exacto, aun cuando el nombre y el número de teléfono que usted desee se encuentren en la base de datos. Esto se debe a que la sentencia IF CLV$ = MODCAM$ es totalmente inflexible; no permite que exista la más mínima diferencia entre la serie de caracteres a que el usuario da entrada en respuesta a la solicitud y la serie de caracteres almacenada en MODCAM$(MID). En una agenda de direcciones corriente, el ojo humano puede ir explorando la página de arriba a abajo y es capaz de tener en cuenta todo tipo de pequeñas diferencias que puedan existir entre la representación verdadera del registro y lo que se está buscando. El ordenador no puede hacer lo mismo.

Existen, sin embargo, algunas formas de evitar esto, aunque éstas suelen implicar un esfuerzo de programación extra y su ejecución lleva algo más de tiempo. La primera mejora sería verificar primero solamente el apellido, y por esta causa resulta coherente que el nombre almacenado en MODCAM$ esté en la forma de APELLIDO (espacio) NOMBRE DE PILA. Previamente, en nuestro curso de programación BASIC (véase el recuadro "Complementos al BASIC"), desarrollamos una rutina para invertir el orden de un nombre; ésta se puede incorporar como una subrutina dentro de la rutina INCLREG cuando se cree el campo MODCAM$.

Habiendo conseguido localizar la primera aparición del apellido requerido, la rutina ENCREG debería entonces verificar la parte del nombre de pila de ese elemento para ver si es idéntica a la entrada del nombre (CLV$). Si lo es, no hay ningún problema: se ha encontrado el registro. Sin embargo, si no es idéntica el problema empieza a complicarse, por lo cual debemos planificar nuestra estrategia con sumo cuidado. Podríamos, por ejemplo, buscar a través de todos los nombres de pila y, de no hallarse ningún emparejamiento exacto, empezar a buscar una pareja aproximada. La dificultad será: ¿qué es, exactamente, una pareja aproximada?

En el programa anterior, en lugar del mensaje "REGISTRO NO HALLADO" podría ser mejor dar un mensaje como "NO HALLADA PAREJA EXACTA, ¿PRUEBO CON UNA PAREJA CERCANA? (¿S/N?)". ¿Y qué significan las palabras "pareja cercana"? ¿Es Paco una pareja cercana de Francisco? ¿Y qué diría de Francis? Estas dos últimas representan posibles entradas para el programa ENCREG. Intentemos definir a qué nos referimos con la expresión una "pareja cercana" y empecemos luego a desarrollar un programa en BASIC para hallar la pareja más cercana para una entrada.

Supongamos que la serie de la memoria era FRANCISCO. ¿Cuál de las siguientes parejas es la más cercana: FRAN o FRNCS? La segunda extrae cinco letras de un total de nueve, mientras que la primera extrae sólo cuatro de un total de nueve. Por otra parte, la primera posee cuatro letras en la secuencia correcta, mientras que la segunda sólo posee dos en dos ocasiones.

Se trata de una elección en gran parte arbitraria. Nosotros optaremos por dar prioridad a una pareja exacta entre CLV$ y una subserie del nombre de la memoria. Si no se hallara una pareja exacta de ninguna subserie, el programa intentaría obtener el mayor número de letras en común. Enunciado en términos de entrada y salida, el programa es:

INPUT
Una serie de caracteres
OUTPUT
La pareja más cercana de la serie entrada

El siguiente programa, en un seudolenguaje bastante aproximado al BASIC, buscará a través de las series de una matriz y examinará las primeras "n" letras de cada una, siendo "n" el número de letras de la clave (CLV$). De no haber ninguna pareja, se imprimirá un mensaje en ese sentido:

```
DIM MAT$(4)
FOR L = 1 TO 4
READ MAT$(L)
NEXT L
DATA "FRANCISCO", "FERNANDO", "FRANÇOISE",
  "FRANCISCA"
LET CLV$ = "FAR"
LET LCLV = LEN(CLV$)
LET BUSCA = 0
LOOP FOR INDEX = 1 TO 4
  IF CLV$ = LEFT$(MAT$(INDEX),LCLV)
    THEN PRINT "LA PAREJA ES   ";MAT$(INDEX)
      LET BUSCA = INDEX
  ENDIF
ENDLOOP
IF BUSCA = 0
  THEN PRINT CLV$; "NO ES PAREJA EXACTA
      DE NINGUNO"
    PRINT "PRIMEROS";LCLV;"CARACTERES"
```

Después de esto, el programa podría seguir adelante mirando los grupos de caracteres de LCLV de longitud, empezando por el segundo carácter de cada serie. Si ninguno de ellos concordara, se podrían buscar los grupos que empezaran por el tercer carácter, y así sucesivamente. Finalmente, si ninguna de las ternas de caracteres de las series concordaran, el programa podría intentar averiguar qué serie tenía el mayor número de letras en común con CLV$. Y esto lo dejamos como un ejercicio para el lector.

En realidad podríamos escribir páginas y páginas acerca del tema de las parejas "libres" y de las diferentes técnicas empleadas en los paquetes de bases de datos comerciales. La mayoría de éstos ofrecen la posibilidad de buscar en algunos de los primeros caracteres del campo, como la codificación que acabamos de desarrollar más arriba. Otros recuperarán un registro si en algún lugar del campo aparece la secuencia de caracteres especificada, o incluso si apareciera en cualquier lugar del registro. La capacidad de "extravagancia" es particularmente útil: al especificar J?S nos encontraríamos con JESUS o JOSE, pero no con JUAN.

Complementos al BASIC

SPECTRUM

Éste es el listado del programa para invertir el orden de nombre de pila y apellido, que se publicara por primera vez en la página 136:

```
100 CLS
200 PRINT "INTRODUCIR NOMBRE EN LA
    FORMA"
300 PRINT "NOMBRE-DE-PILA APELLIDO"
400 PRINT "P. EJ. MARIA PEREZ"
500 INPUT "DE ENTRADA AL NOMBRE";N$
600 GOSUB 9500
700 PRINT "EL NOMBRE EN LA FORMA
    ESTANDAR ES"
800 PRINT N$
1000 STOP
9500 REM S/R PARA INVERTIR ORDEN DEL
     NOMBRE
9520 GOSUB 9600
9540 IF P = 0 THEN RETURN
9560 LET N$ = A$+ ",   "+P$
9580 RETURN
9600 REM S/R PARA CORTAR N$ POR UN
     ESPACIO
9620 LET N = LEN(N$)
9630 LET P = 0
9640 FOR K = 1 TO N
9650 IF FN M$(N$,K,1) = "   " THEN LET
     P = K: LET K = N
9660 NEXT K
9670 IF P = 0 THEN RETURN
9680 LET P$ = FN L$(N$,P-1)
9700 LET A$ = FN R$(N$,N-P)
9720 RETURN
9900 REM FUNCIONES EN SERIE
     DEFINIDAS POR EL USUARIO
9990 DEF FN M$(X$,P,N) = X$(P TO
     P+N-1)
9991 DEF FN L$(X$,N) = X$(   TO N)
9992 DEF FN R$(X$,N) = X$
     (LEN X$-N+1 TO   )
```

LEFT$

En el Commodore 64, Vic-20, Oric-1 y Lynx, reemplazar desde la línea 9600 a la 9720 del listado para el Spectrum por las líneas siguientes:

RIGHT$

MID$

```
9600 REM S/R PARA CORTAR N$ POR UN
     ESPACIO
9620 LET N = LEN(N$)
9630 LET P = 0
9640 FOR K = 1 TO N
9650 IF MID$(N$,K,1) = "   " THEN LET
     P = K: LET K = N
9660 NEXT K
9670 IF P = 0 THEN RETURN
9680 LET P$ = LEFT$(N$,P-1)
9700 LET A$ = RIGHT$(N$,N-P)
9720 RETURN
```

y suprimir desde la línea 9900 hasta la 9992.

INSTR

En el Dragon 32 y en el BBC Micro, reemplazar desde la línea 9600 hasta la 9720 del listado para el Spectrum por las líneas siguientes:

```
9600 REM S/R PARA CORTAR N$ POR UN
     ESPACIO
9620 LET N = LEN(N$)
9640 LET P = INSTR(N$,"   ")
9670 IF P = 0 THEN RETURN
9680 LET P$ = LEFT$(N$,P-1)
9700 LET A$ = RIGHT$(N$,N-P)
9720 RETURN
```

y eliminar desde la línea 9900 a la 9992.

Como ya hemos mencionado anteriormente, INSTR es una función muy útil, en especial al tratar con aplicaciones de tipo base de datos como lo es ésta. Si su máquina dispone de INSTR, tal vez se decida a intentar una forma más sofisticada de emparejamiento "libre".

INPUT

En el BBC Micro, reemplazar la línea 500 del listado para el Spectrum por:

```
500 INPUT "DE ENTRADA AL NOMBRE",N$
```

Tony Lodge

Konrad Zuse


Cortesía del Siemens-Museum, Munich

Zuse logró en Alemania resultados similares a los obtenidos por Von Neumann en Estados Unidos

Con frecuencia en distintos lugares del mundo se producen inventos simultáneamente a partir de ideas que surgieron y se desarrollaron independientemente una de otra. En la década de los cuarenta, mientras en Estados Unidos se estaba construyendo el primer ordenador de válvulas (el ENIAC), un ingeniero alemán, Konrad Zuse, trabajaba en una calculadora programable, que se considera como el primer ordenador de la historia.

Zuse nació en Berlín el 22 de junio de 1910. Después de estudiar en la Universidad Técnica de la ciudad, trabajó como ingeniero aeronáutico para la Henschel Aircraft Company, dedicándose al diseño de alas. Los principios matemáticos básicos aplicados al refuerzo de las alas de los aviones para resistir las tensiones del vuelo a alta velocidad ya se habían establecido en la década de 1920. No obstante, los cálculos individuales necesarios para producir cada par de alas requerían equipos de personas trabajando con máquinas de calcular mecánicas y reglas de cálculo. Zuse comprendió muy pronto la necesidad de contar con una máquina que pudiera efectuar con rapidez este trabajo que ocupaba tanto tiempo. Por las tardes, en compañía de otros amigos, emprendió, en el piso de sus padres, la labor de construir un ordenador que pudiera realizar esta tarea.

Su primera máquina, el Z1, era un dispositivo mecánico que podía efectuar las cuatro operaciones aritméticas elementales, calcular raíces cuadradas y convertir números decimales a notación binaria y viceversa. Aunque no estaba enterado de los logros de Charles Babbage (véase p. 220), cuyo ingenio diferencial se había creado para efectuar los laboriosos cálculos que requerían las tablas náuticas, Zuse había llegado a conclusiones muy similares y a otras que eran mucho más avanzadas. El descubrimiento más sensacional de Zuse se produjo al comprobar que una palanca era un interruptor que se podía colocar en una de dos posiciones (encendido o apagado) y que, por consiguiente, se podía utilizar ya como medio para almacenar datos, ya como dispositivo de control.

Zuse pretendía representar tanto los datos como las instrucciones en forma binaria, y el año 1941 inició la construcción de un ordenador electromagnético, al que él llamó Z2. Dedicado de lleno al esfuerzo de la guerra, el gobierno alemán se mostró poco interesado, al principio, en el invento. No obstante, finalmente acabó por reconocer el potencial interés militar del aparato y le proporcionó fondos a Zuse para desarrollar el Z3. Éste habría de ser un ordenador eléctrico, con un tendido de cables eléctricos, que posibilitaron un diseño más compacto y elegante, en lugar de los enlaces mecánicos que utilizó en las máquinas anteriores.

Zuse construyó el Z3 pese a no pocos contratiempos. Los bombardeos de Berlín por los aliados le obligaron a trasladar su taller en diversas ocasiones. Dos veces lo llamaron a filas, sólo para mandarlo de vuelta desde el frente oriental para que continuara su trabajo. La escasez de materiales durante la guerra le forzó a improvisar, obligándolo a servirse de piezas extraídas de los engranajes de conmutación telefónicos y a utilizar copias de antiguas películas, perforadas con códigos de ocho agujeros por fotograma, en lugar de cinta de papel.

El Z3 podía almacenar 64 palabras, cada una de ellas de 22 bits de longitud. A la información se le daba entrada a través de un teclado y los resultados se exhibían visualmente en un conjunto ordenado de lámparas montadas sobre un tablero. Lamentablemente, el Z3, al igual que todos los ordenadores anteriores de Zuse, fue destruido en 1945 durante un bombardeo de Berlín.

Uno de los ordenadores lo adaptó la Henschel Aircraft Company para ayudar en la construcción de la bomba volante HS-293. Se trataba de un avión no tripulado que se lanzaba desde un bombardero y se guiaba hasta su objetivo por radio.

El último ordenador que produjo Zuse durante la guerra, el Z4, había incrementado la longitud de sus palabras a 32 bits. Cuando los aliados se acercaban a Berlín, la máquina fue trasladada a Gotinga. Finalmente quedó instalada en Basilea (Suiza), donde estuvo en funcionamiento hasta 1954.

Zuse no consiguió fabricar ordenadores en la Alemania de la posguerra, por lo cual dedicó sus esfuerzos a la teoría informática. Desarrolló un sofisticado lenguaje denominado Plankalkül que podía tratar lógicamente tanto con matemáticas como con información más general. Cuando pudo volver al campo de la creación de ordenadores, fundó la Zuse Company, que fue la fábrica de ordenadores más importante de Alemania hasta 1969, en que fue absorbida por Siemens Corporation.

La informática en la guerra
Los ordenadores de Zuse se desarrollaron para sustituir a los equipos de técnicos que efectuaban cálculos aeronáuticos trabajando con reglas de cálculo. Se aplicaron especialmente en el diseño de las bombas volantes V1 y V2 (en la fotografía), que utilizaría Alemania al final de la segunda guerra mundial


© The Imperial War Museum

PARA JUGAR A LO GRANDE
(INSTANTANEAMENTE)

Presentamos **el Interface 2 ZX** Pensado y diseñado por SINCLAIR para unirse a la perfección con tu microordenador Spectrum.

Si a la hora de elegir tu microordenador optaste por el mejor, es lógico que elijas ahora el Interface 2 ZX

Ya habrás podido deleitarte con la más amplia variedad de juegos existentes para tu Spectrum (la más extensa del mercado). Ahora con el Interface 2 ZX vas a tener más ventajas para tu Spectrum:



- Podrás conectar Joysticks para sacarle, aún, mayor rendimiento a tus mejores juegos y divertirte con aquellos exclusivamente disponibles en **Cartuchos ZX:** correr, saltar, volar... a lo grande.
  ¡Menuda diferencia!
- Además, al ser cartuchos con memoria ROM, podrás, con tu SPECTRUM de 16 K, jugar con programas hasta ahora reservados para 48 K, sin ampliar la memoria.
  ¡Vaya ahorro!
- Al conectar el Interface 2 ZX tienes la certeza de poseer un periférico pensado por SINCLAIR para SINCLAIR. Tu microordenador queda a salvo de circuitos poco fiables.
  ¡Un alivio!
- Al adquirir el Interface 2 ZX y los Cartuchos ZX en la red de Concesionarios Autorizados, podrás exigir la tarjeta de garantía INVESTRONICA, única válida en territorio nacional.
  ¡Una tranquilidad!

Interface 2 ZX y Cartuchos ZX

Si aún no los tienes
no sabes lo que te pierdes

Solicita una demostración en cualquier Concesionario Autorizado INVESTRONICA.



CENTRAL COMERCIAL: Tomás Bretón, 60
Tel. 468 03 00 Telex: 23399 IYCO E Madrid.
DELEGACION CATALUÑA: Camp, 80 - Barcelona - 22

UNION PERFECTA

Así se comportan los periféricos creados por SINCLAIR para SINCLAIR: de forma perfecta. Y es lógico.

Cada vez que SINCLAIR diseña un microordenador, no lo hace de una manera aislada. Simultáneamente crea todos esos periféricos que van a hacer más potente, preciso y útil el microordenador que tiene entre manos.

Periféricos pensados y diseñados para dar un servicio óptimo, pero con un precio razonable, dentro de la filosofía SINCLAIR:

"Hacer la informática accesible a todos".

Por eso cuando creó el ZX 81 vio la necesidad de dotarlo con una ampliación de memoria de 16K RAM para que no quedara pequeño y de una impresora sencilla y barata pero útil y precisa.

Así es la filosofía SINCLAIR. Así son los periféricos de SINCLAIR para SINCLAIR.



